Color Image Segmentation

THESIS

Kimberley A. McCrae
Captain, USAF

AFIT/GEO/ENG/93D-02

AFIT/GEO/ENG/93D-02

Color Image Segmentation

THESIS

Presented to the Faculty of the Graduate School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Kimberley A. McCrae, B.S.E.E.
Captain, USAF

December, 1993

Acknowledgements

I'd first like to thank my committee, Captain Dennis Ruck and Doctors Steven Rogers and Mark Oxley; with special thanks to Matt Kabrisky for several interesting conversations. My last faculty thanks goes to nearly-doctor Tom Burns for all his help and many ideas.

Big thanks to the Latex masters, Curtis Martin, Neale Prescott, and John Keller, they probably saved the life of at least one Next terminal. I'd also like to acknowledge the other members of the pattern rec group, Martin Chin, Bob MacDonald, and Gary Shartle for making this an overall enjoyable experience.

My last thanks goes to my husband, Jack (the brightest boy in school), for making me turn-off the football game and write this thing.

Kimberley A. McCrae

Table of Contents

List of Figures

List of Tables

AFIT/GEO/ENG/93D-02

Abstract

The most difficult stage of automated target recognition (ATR) is segmentation. Current AFIT segmentation problems include faces and tactical targets; previous efforts to segment these objects have used intensity and motion cues. This thesis develops a color preprocessing scheme to be used with the other segmentation techniques. A neural network is trained to identify the color of a desired object, eliminating all but that color from the scene. Gabor correlations and 2D wavelet transformations will be performed on stationary images; and 3D wavelet transforms on multispectral data will incorporate color and motion detection into the machine visual system. The thesis will demonstrate that color and motion cues can enhance a computer segmentation system. Results from segmenting faces both from the AFIT data base and from video taped television are presented; results from tactical targets such as tanks and airplanes are also given. Color preprocessing is shown to greatly improve the segmentation in most cases.

Color Image Segmentation

I. Introduction

1.1 Background

Automated image processing is a useful tool for quickly evaluating a great deal of data, but machines are not yet as competent as people. Machine image processing has many military and civilian applications including target identification from satellite imagery, terminal guidance of smart bombs, user authentication for security systems, tumor identification from CAT scans, and blood cell counts from stained microscope slides. Unfortunately, machines do not process images as accurately as humans.

Usually machines only process intensity information (how much light is reflected off an object); they are easily fooled if there is low contrast between an object and the background. Humans, on the other hand, take advantage of more information, they have four parallel vision systems, one for color, one for motion, one for form (shape) and color, and one for form and motion (19). People are not easily fooled, if the intensity information is misleading, humans simply fall back on color (in bright light) or motion cues to determine what objects are present.

If machines could use color and motion cues, like humans, their performance could be greatly enhanced. A machine could find camouflaged tanks, indistinguishable in intensity data, when the tanks move, or a machine could locate faces in a crowd based on color then feed the position to a face recognition routine. Machines could even surpass human performance as they can "see" beyond the human visual range. Current detector technology can collect data well into the infra-red (IR) wavelengths, which is well beyond human capabilities. The machine would consider this data as just another color and process it as color.

1.2 Problem

This thesis will emulate the human visual system by incorporating color preprocessing into image segmentation techniques. Gabor correlations and 2D wavelet transformations will be performed on stationary images; and 3D wavelet transforms on multispectral data will incorporate color and motion detection into the machine visual system. The thesis will demonstrate that color and motion cues can enhance a computer segmentation system similarly to how these cues enhance human visual systems.

1.3 Scope

This thesis will investigate color segmentation as a preprocessing step for other object recognition research. Images of faces, tanks, and airplanes will be segmented using a combination of a neural network color segmenter and wavelet transformations or correlations, the results of color segmented images and non-color segmented images will be compared.

1.4 Assumptions

This thesis assumes the color of the desired object is known, i.e. there are samples of the target available for training. This work also assumes the target's range is known. This will determine the size of the object in the image, which will indicate the wavelet decomposition level. It will also indicate what size Gabor correlator to use.

1.5 Approach

Color preprocessing techniques will be investigated to determine if an Linde, Buzo and Gray (LBG) clusterer or a multilayer perceptron is more effective. The better of these methods will be used to segment faces, tanks, and airplanes from images and sequences of images. These segmented images will be further segmented by a wavelet transformation and a Gabor cross correlation. A sequence of images containing moving airplanes will be segmented by a 3-dimensional wavelet transformation, incorporating color and motion to find the planes.

1.6 Conclusion

Color preprocessing is potentially an important tool for image segmentation. The following chapters provide a demonstration of its usefulness. Chapter II discusses common color segmentation techniques, as well as describing the tools (such as LBG clustering, multilayer perceptrons, and wavelet decomposition, used in this thesis. Chapter III presents the methodology for this work, Chapter IV discusses the results, and Chapter V provides a summary and some thoughts on future work.

II. Literature Review

2.1 Introduction

Color image processing has been gaining interest in recent years. There are many issues to be considered including what color representation system to use and what type of classifier is most effective. Most techniques use the red, green, blue (RGB) system and process each set of components as if it were a gray scale image. The most common classifiers are variations of different clusterers or variations of histogram methods. In the following sections typical color segmenters are presented as well as background information on the form segmentation techniques used in this thesis.

2.2 Color Segmentation

The first step in developing a color segmentation scheme is to determine in which color coordinate system to work. Most techniques use either the RGB or the hue, saturation, intensity (HSI) systems. RGB is popular because it is how machines acquire color; HSI is popular because it is similar to how humans perceive color (12). In addition to these, several researchers have come up with a system specific to their application (10) (17) (4). The tougher question is what to do with the color data once it is obtained.

A common histogram technique for determining a coarse segmentation is scale-space filtering (8) (3). In this method the histogram of a color component (r, g, or b) is convolved with several scaled gaussians; this smooths the histogram, the larger the gaussian the smoother the result. The peaks and valleys of the histogram are found by taking the second derivative and looking for the zero-crossings. A map created from the zero-crossings of all the scales is called a fingerprint and by examining this fingerprint the colors present in the image can be determined. Usually a second stage of processing (such as Markov random fields) is used for fine segmentation.

Huang reported good success with scale space filtering particularly in determining the number of clusters needed to segment the image, he did require a second stage of processing to obtain reasonable results (8).

Another clustering technique based on a histogram was presented by Celenk (4). He used a variation of the HSI coordinate system and histogrammed each component looking for the most prominent peak in any of the three histograms. Once he had determined the most prominent peak, he projected only that portion of the color space onto either of the other two coordinate axes and looked for the peak again; he then projected only that peak onto the third coordinate, the peak there completed the determination of the color volume for that cluster. He then ignored that cluster and repeated the procedure with the remaining data. Unfortunately, the article was printed in gray scale, making it difficult to visually determine how well this technique worked.

2.3 Clustering

Two clustering techniques are used in this thesis, the first is the Linde, Buzo, and Gray (LBG) method, the second is a simple heuristic method. Linde, Buzo, and Gray's "An Algorithm for Vector Quantizer Design" (13) is the paper which describes their vector quantization scheme. The LBG algorithm is now widely used and is the standard of comparison for other techniques. In the article they describe how to perform LBG clustering and get the prototypes for each partition, and they give a few examples of the LBG's performance.

The straight-forward approach is to iteratively cluster the data, find the prototypes, then recluster the data until some convergence is reached. LBG describe a splitting technique where the first prototype is simply the average of all the data. That prototype is then split - two new prototypes, each a specified distance from the original, are created and the data is clustered about these two positions based on a minimum distance metric. The prototypes for these two clusters are found by averaging the data within each partition. The algorithm is repeated by splitting each of these prototypes. The process continues until either the total error (distance from all data points to their prototypes) is below some convergence criteria or a preset maximum number of clusters is reached.

The authors presented three examples of using this algorithm. The first example used zero mean unit variance Gaussian random variables; the second used a noisy signal; and the third was a speech signal. In all three cases they achieved nearly optimal (minimum distortion) clustering; they reserved the proof for a different paper (7).

The second clustering scheme used in this thesis is "a simple cluster-seeking algorithm" described by Tou and Gonzalez in their book ***Pattern Recognition Principles*** (18). This technique is truly brute strength and ignorance (my favorite). A data point is taken to be the first cluster, and a threshold is chosen. If the distance from the cluster prototype to the next data point is less than the threshold the data point is incorporated into the cluster. If the distance is greater than the threshold, this data point is taken as a new cluster prototype. If the distance from the next data point to either of these prototypes is greater than threshold, this data point becomes a new cluster prototype; otherwise, it is incorporated into the cluster it was closest to. The process continues until all the data points are incorporated into clusters. The advantage of this technique is that it is quick and easy. However, it is sensitive to choices of initial cluster center, threshold, and the order of data presentation.

2.4 Wavelets

The basis for wavelet decomposition is in "A Theory for Multiresolution Signal Decomposition: the Wavelet Representation" by Stephane Mallat (14). This paper gives the mathematical development of wavelet analysis, explains how to extend the process to two-dimensions, and gives a couple of examples.

The idea behind the multiresolution analysis is to give both time and frequency (or space and spatial frequency) information. A Fourier transform will give frequency information, but it won't indicate where in the signal a given frequency is present. The wavelet transform will give an indication of what frequencies are present and where they are located. The resolution in time is inversely proportional to the resolution in frequency; i.e., a wavelet can determine that a narrow frequency band is present somewhere in a large time segment, or conversely, it can determine that some part of a wide frequency band is present in a small time segment.

The wavelet decomposition can be thought of as a quadrature mirror filter, where the signal is convolved with a lowpass filter then downsampled to create an approximation, and it is also convolved with an orthogonal bandpass filter then downsampled to create a detail signal. The details are the difference between the original signal and the approximation such that the orthogonal sum of the approximation and the details returns the original signal. At this point the high frequency components of the original signal have been taken out leaving a slightly fuzzy signal as the approximation; the details contain the high frequency components so they will show where sharp transitions are present in the signal. The lowpass and bandpass filters are then dilated and the approximation is filtered yielding a coarser approximation and another level of details; now the original input can be reconstructed by the orthogonal sums of the coarse approximation and the two levels of details. The approximation is even fuzzier and is perhaps missing small segments (high frequencies), the details will contain these small segments.

The wavelet decomposition can be extended to two-dimensions quite easily. An image is filtered and the columns are downsampled, then these approximation and details are filtered again and the rows are downsampled. This yields one approximation and three sets of details, one corresponding to vertical frequencies, one corresponding to horizontal frequencies, and one corresponding to both. These details will determine the position of vertical edges, horizontal edges, and diagonals and corners.

Mallat gives two pertinent examples, one is a simple square, the other is a textured image composed of little tilted L's and little untilted L's. In both cases the wavelet decomposition works as advertised. For the square, one set of details shows the vertical edges, another shows the horizontal edges, and the last shows the corners. For the textured image the untilted L's show-up in the vertical and horizontal details, while the tilted L's show-up in the diagonal details.

Tom Burns took this two-dimensional decomposition and extended it to three dimensions, the third dimension being time; this allowed him to look for motion over several frames of a sequence (2) (1). This extension is similar to the previous one done by Mallat. The approximation and three sets of details are filtered again and the frames are downsampled to yield an approximation and seven sets of details corresponding to fast and slow moving vertical and horizontal edges, fast and slow moving corners, and high temporal frequencies. Burns also showed that the vertical, horizontal, and temporal filters need not be identical; this allowed him to use filters which fit the signal on each axis.

2.5 Multilayer Perceptron

Steve Rogers presents the fundamentals of a multilayer perceptron (MLP) in his book *An Introduction to Biological and Artificial Neural Networks* (16). The idea is that a network can learn to be a nonlinear classifier. Typically there are three layers in an MLP, an input layer, a hidden layer, and an output layer. Each input is multiplied by a set of weights (one weight for each hidden node) then summed with the other weighted inputs and a bias term at each hidden node. A nonlinear transformation (usually a sigmoid) is performed on the sum to produce the output. The learning takes place by updating the weights. An input vector is fed through the network. If the output doesn't correspond to the desired class an error correction (gradient descent) is propagated back through the network updating the weights.

The only paper found using neural networks for color analysis was by Nakauchi, Nakano, and Usui (15). They were trying to simulate how the brain takes physical color information and transforms it into psychological perceptions of color. They used a five layer network with 81 input nodes, 10 nodes with sigmoid output for the first hidden layer, 3 nodes with linear output for the second hidden layer, 10 nodes with sigmoid output for the third hidden layer, and 81 output nodes. Three nodes were chosen for the center layer as three is the fundamental number for color vision (red, green, blue; hue, saturation, intensity). Spectral data from Munsell color chips were taken over the visible range at $5nm$ increments to get the input feature vectors; the network was trained using 1280 data sets and tested on 289 data sets.

The purpose of this experiment was not so much to correctly classify an input data vector, but to determine what the three central nodes responded to; i.e., they were trying to reduce their 81 element feature vector to a three element feature vector which represented the optimal feature set for Munsell color data. They found that one node responded to value (intensity), and the other two nodes responded to hue and chroma, one to red - green and the other to yellow - purplish blue. These three features correspond the Munsell color solid organized according to psychological attributes. The authors conclude that the network was able to learn the optimal feature set for color, and that a five-layered neural network "has an excellent ability for elucidating the internal representation of the neural system" (they must mean neural network neural system and not human neural system).

2.6 Conclusion

The following chapters will present a color segmentation method using an MLP. The results of this segmentation will be combined with a Gabor correlation and a wavelet transformation to demonstrate the utility of color segmentation.

III. Methodology

3.1 Introduction

Color image segmentation was used as a preprocessor to conventional segmentation filters. The following sections describe the data sets used in these experiments, a heuristic method for finding eyes in a color segmented image, a wavelet decomposition method for finding objects in scenes, and a Gabor correlation method for finding the objects.

3.2 Data Sets

Three sets of data were used in these experiments; faces, tanks, and airplanes. All the data were collected using a VHS video camera. The faces were taken by Dennis Krepp, Kevin Gay, and John Keller for their face recognition theses (11) (6) (9). Most faces were basically still with plain backgrounds; a typical example is shown in Figure 1. The tank data were collected by Ken Fielding for his dissertation (5). These tanks were stationary; an example is shown in Figure 2. The final set of data was a sequence of airplanes collected by Greg Powers of the Avionics Laboratory. This was taken with a constant background as a plane took-off and landed; the color quality of this tape was not very good as shown in Figure 3. The sequence consisted of 32 frames, Figure 3 shows the first, last, and a couple intermediate frames.

The data was digitized into the computer using the `vfctool` frame grabber on the Sun Sparc station II computer. The stationary pictures (faces and tanks) had several frames of each object; the data were easily captured by playing the tape while observing in `preview` mode and `grabbing` the desired image. The moving data had to be transferred to an 8mm video cassette to be played on a VCR with single frame advance. Grabbing these frames was tedious as each frame had to be held still by the VCR while the computer grabbed and saved it. The data were saved as Sun

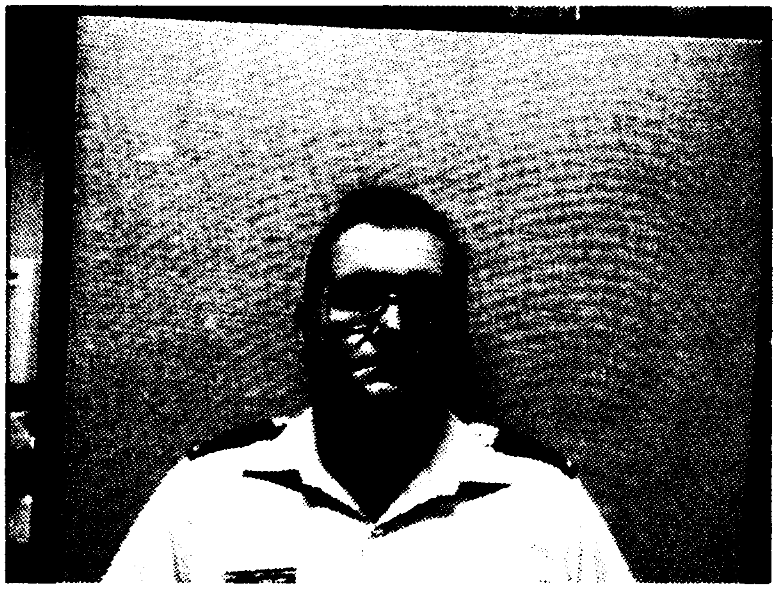


Figure 1. Typical face image used in color segmentation.

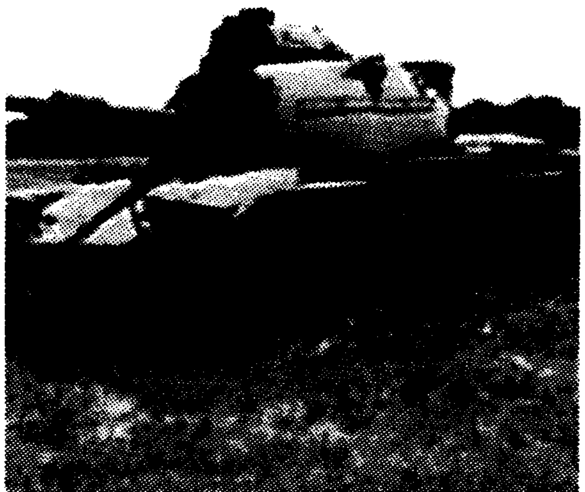


Figure 2. Typical tank image used in color segmentation.

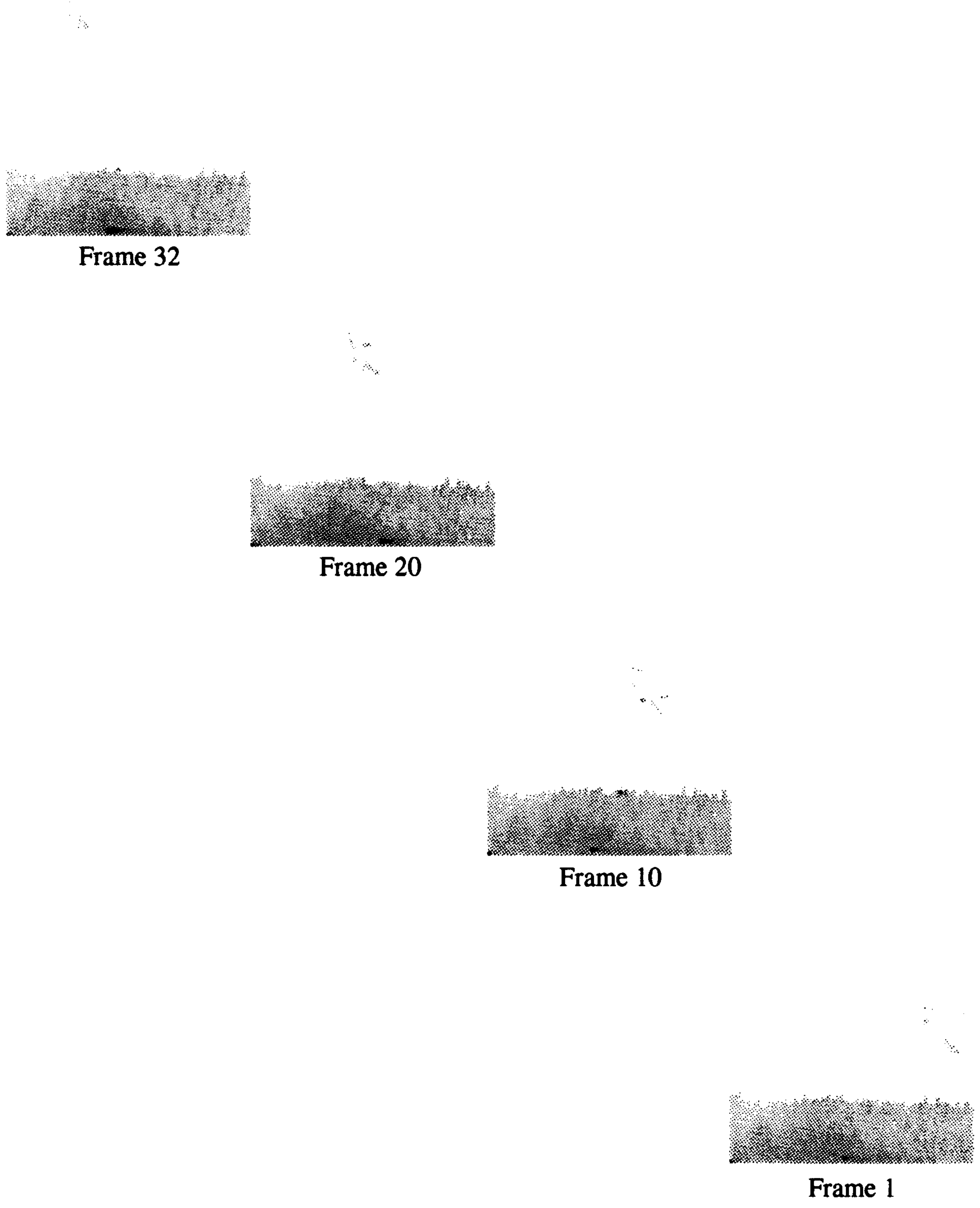


Figure 3. Typical airplane images used in color segmentation.

Raster files in 24-bit RGB format. This is a binary format with a 32 byte header consisting of the the information shown in Table 3.2.

Header Variable	Typical Value
Magic # for Sun Raster	1504078485
Number of pixels in the x dimension	640
Number of pixels in the y dimension	480
Number of bits per pixel	32
Total number of bytes in the picture	1228800
Don't know	1
Don't know	0
Don't know	0

Table 1. Typical Sun Raster 24-bit RGB format header.

Each pixel was represented by 32 bits, the first eight bits were a buffer, the second eight corresponded to the blue value, the third eight corresponded to the green value, and the final eight corresponded to the red value. This was the easiest format for reading into a C-program and extracting the red, green, and blue values. C-code for reading and writing this format is in Appendix A. Sections of any image could be cropped-out by using the `Grab` application available on the Next computers. Using "grab selection" any rectangular portion of a picture could be copied into a new image and saved in the TIFF format. The TIFF image had to be loaded into **`vfctool`** and saved as Sun Raster 24 bit RGB to be compatible with the C-code developed for this thesis. A second cropping method was the C program reduce.c which was useful for cropping specific sizes and resolutions. This program allowed a user to determine the size of the cropped section, as well as the resolution, and the position it was taken from. This way a user could convert a 256 × 256 portion of the original image from anywhere in the image into a 128 × 128 or a 64 × 64 image.

3.3 LBG Clustering Segmentation

The ESPS signal processing package was used to LBG cluster the pixels of face images. The input vector was the red, green, and blue (RGB) values of each pixel. ESPS vector quantized the image into 32 clusters. The ESPS commands are given in Appendix B. Several complete pictures (face and background) were used to develop the codebook, then several sections of just face or just background were vector quantized to be used in training a multilayer perceptron (MLP) look-up table. The MLP look-up table was developed in the LNKnet software package available on the Sparcs. The network input vector contained 32 elements, all zeros except for a one in the place corresponding to the codebook value of the pixel. There were two network outputs, "face" and "not face".

Once the codebook was developed and the network trained, pixels from additional face images were fed through the ESPS and LNKnet software to be quantized and classified as "face" or "not face". The images were then reconstructed by the C program `reconstruct.c` using only the face pixels.

3.4 Multilayer Perceptron Segmentation

In this approach the LBG clustering step was omitted and the RGB pixel values were used as the input vector to the LNKnet mlp program. Three different networks were trained, one for faces, one for tanks, and one for airplanes. The networks learned which combinations of RGB values corresponded to object and which combinations corresponded to background. All three networks had 25 hidden nodes (the default of the LNKnet MLP algorithm); the face network was trained with 11,068 "face" and 30,109 "not face" pixels, the tank network was trained with 10,908 "tank" and 10,860 "not tank" pixels, and the airplane network was trained with 1761 "plane" and 14,297 "not plane" pixels. The MLPs were trained for 15 epochs (also the default) achieving training errors of 7.69%, 7.06%, and 3.60% for face, tank, and airplane, respectively. As above, once the network was trained several images were fed through the system to be classified as "object" or "not object"; and only object pixels were used in the reconstruction.

3.5 Heuristic Eye Finder

Once faces were found it was suggested that an eye finder would be very useful for the face recognition routines developed at AFIT. To do this the neural network was retrained with eyes classified as "not face", this left holes in the reconstructed faces where the eyes used to be, then several methods were used to find the holes.

The first method attempted was highly heuristic. The faces were assumed to be basically centered in the image, this allowed the bottom half of the image to be ignored (with the obvious assumption that no one was standing on his head). Each line of the top half of the reconstructed image was scanned looking for blank segments with lengths anywhere from 10% to 40% of the distance from the first face pixel on that line to the last. If such a segment was found, the coordinates of the center of that segment were saved and clustered with the centers of other segments to determine eye position.

The clustering routine is based on a simple clusterer in *Pattern Recognition Principles* (18). An initial cluster center is set at (0,0), then the Euclidean distance from the position of each line segment to the cluster center is determined, if the distance is less than some threshold (1/10 the picture size in this case) the position is added to the cluster and the center of the cluster is recalculated. If the distance is greater than the threshold a new cluster is created with its center at the position of the segment. The two eye positions are determined to be the centers of the two clusters with the most elements.

3.6 Wavelets

This approach used the wavelet codes developed by Tom Burns for his dissertation (1). A 2-dimensional wavelet was used on the stationary faces and tanks, and a 3-dimensional wavelet was used on the moving airplanes. For comparison the decompositions were performed both on the original image and the color segmented image.

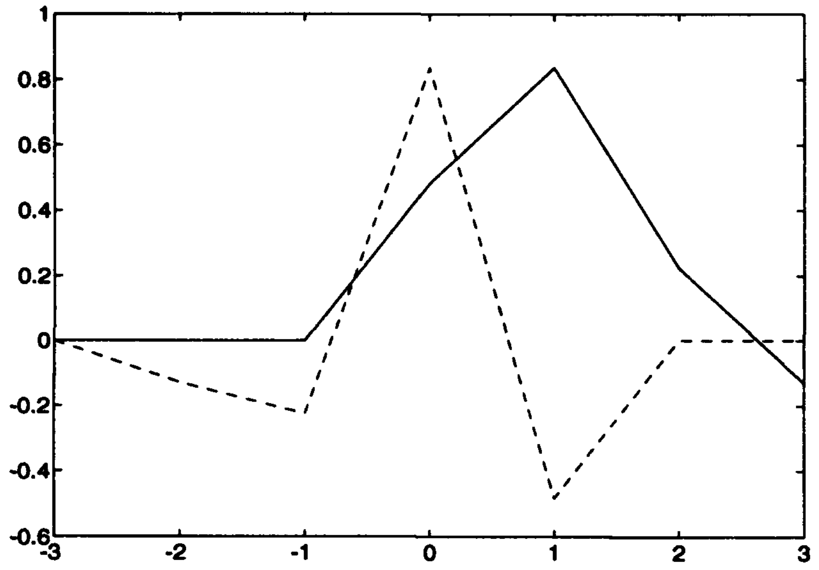
3.6.1 2-D Wavelet. A Daubechies 4-tap wavelet was used for both the x and y decompositions; Figure 4 shows the filter and its Fourier Transform. Here the wavelet is being used for a blob detector, so the eyes of a face must be blobs rather than holes; to achieve this the face's negative is reconstructed, that is, everywhere a face pixel should be there is nothing, and everywhere nothing should be there is a white pixel.

It is assumed that the range of the object is known; therefore, its size in the image is known. This indicates what level of the decomposition to look for the object in. To run the wavelet code the input image size must be a square with a power of 2 as the length of a side; also, the object of interest must be big enough for the wavelet to find it. The face images were fit into this square using the reduce.c program; a 256 × 256 section of the original 640 × 480 image was cropped out and subsampled to a 128 × 128 image, this made a typical eye about 30 × 25 which should be found at the 3rd or 4th level of decomposition. The tanks were about 70 × 30 so they also should have been found at the 3rd or 4th level of decomposition.

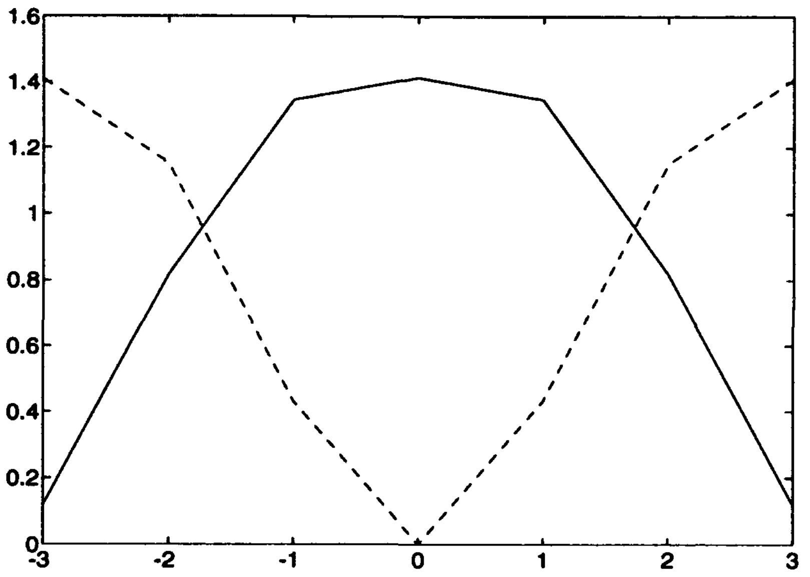
3.6.2 3-D Wavelet. A 3-dimensional wavelet decomposition was performed on the airplane data where there were x, y, and time information available. Daubachies 4-tap filters were used on all three dimensions. Decompositions were performed on both the gray scale image of the original scene and on the gray scale image of the color segmented scene. The airplane data was cropped to 128 × 128 by the reduce.c program. This makes the airplane about 10 × 10 pixels. 32 frames of data were available. It was expected that the plane would show up in the 2nd level spatial decomposition and the 1st level temporal decomposition.

3.7 Correlation

A Gabor filter was developed to correlate with the segmented color images to find the objects of interest. As in the wavelet method the Gabor is a "blob"



Daubechies 4 tap filter (lowpass - solid, bandpass - dashed)



Frequency response of the Daubechies 4 tap filter

Figure 4. Daubechies four tap filters and their Fourier transforms.

detector, therefore, the negative face images were used here as well. Also, as in the wavelet approach, the grey scale image of the original unsegmented color images were correlated with the Gabor for comparison. The Gabor was developed using the following equation:

$$gabor(x,y) = \exp\left[\frac{-\pi((x-m_x)^2+(y-m_y)^2)}{\sigma^2}\right]\cos 2\pi f_x(x-m_x)\cos 2\pi f_y(y-m_y).$$

where m_x, m_y are the centers of the filter, σ is the standard deviation, and f_x, f_y are the cosine frequencies in the x and y directions respectively. These parameters can be adjusted to match the size of the filter to the expected size of the object. The Gabor correlation was performed by the `xcorr2` command of `Matlab`.

3.8 Conclusion

Color preprocessing will be performed on face, tank, and airplane data and incorporated into several second stages of segmentation including a heuristic method, a wavelet transform method and a Gabor correlation method. All of these techniques are for finding the position of objects in scenes. In the following chapter the results of the various methods are presented and compared to determine whether or not color preprocessing helps any of the segmentation techniques.

IV. Results

4.1 Introduction

This chapter presents the results of the various segmentation techniques on both color preprocessed images and the gray scale version of the same images. The following two sections discuss the results of the vector quantization color segmenter for face images, and the neural network color segmenter for face, tank, and airplane images. The remaining sections discuss the results of the object locating techniques.

4.2 Clustering

The ESPS clustering routine was trained using face images (with background) of Duane, Jamie, Neale, and Steve A. (shown in Figure 5). Figure 6 shows "face" and "not face" pixels in RGB space. These plots show this to be fairly linearly separable problem. The MLP look-up table was trained using just face sections of those four identified to LNKnet as class "0" and using just background sections from several images identified to LNKnet as class "1". After the training was complete several images were run through the system for segmentation. Segmented images were reconstructed from only the pixels identified as "face". Six representative examples are shown in Figure 7. Note that Bryon, Craig, and Neale wear glasses while the other three do not; glasses had no effect on the segmentation.

This segmentation technique did a good enough job that people who knew these individuals were able to recognize them from the reconstructed images; however, there are large portions of face missing in all the reconstructed images. Glint off foreheads seemed to be a particular problem (one would be surprised how hard it is to find an Air Force officer with a full head of hair).

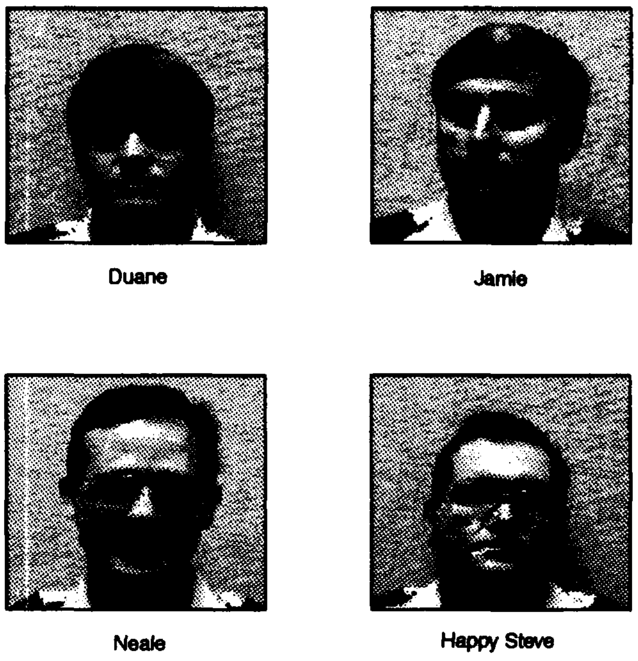


Figure 5. The faces used to develop the LBG vector quantization codebook.

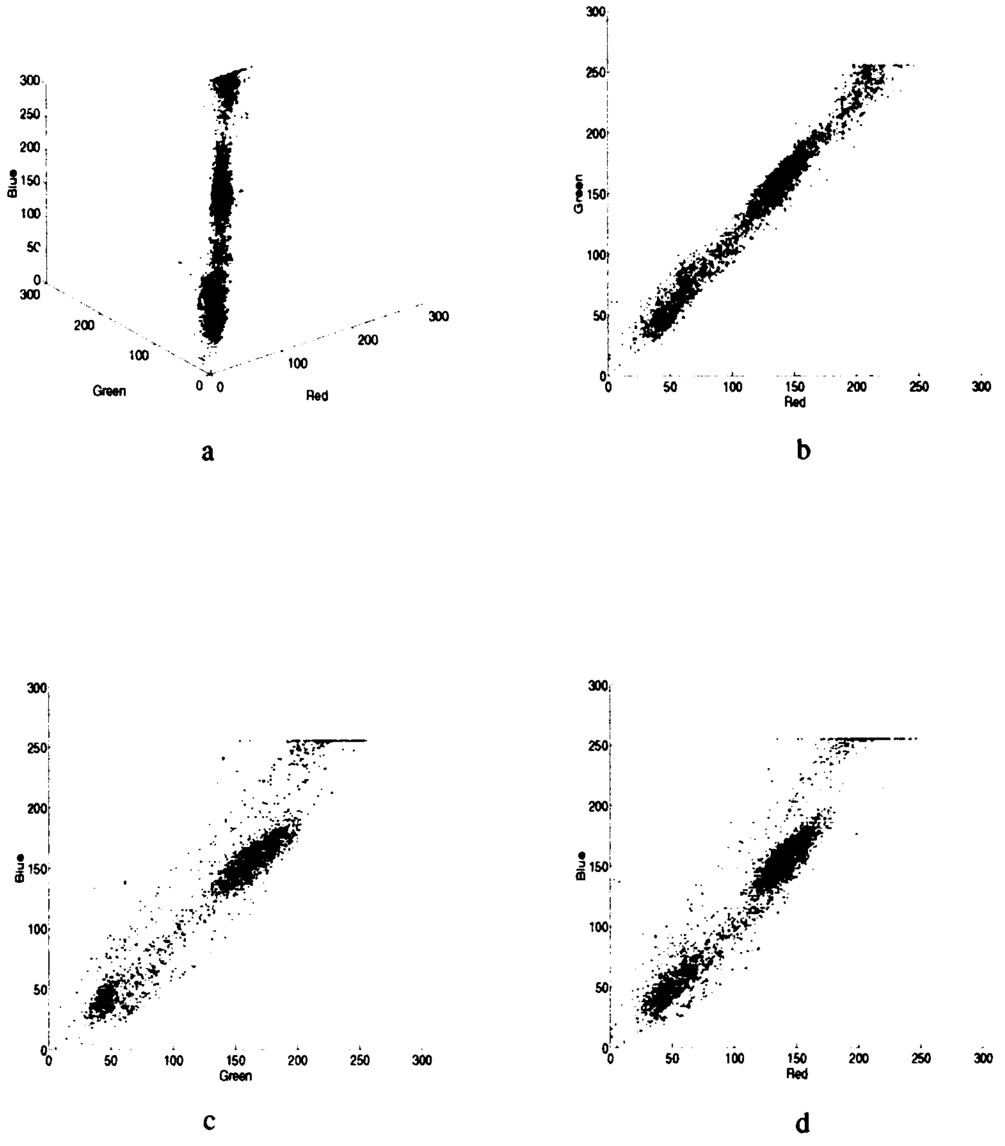


Figure 6. "Face" (yellow) and "Not Face"(red) pixel distribution in RGB space. (a) is a 3-D plot. (b)-(d) are projections.

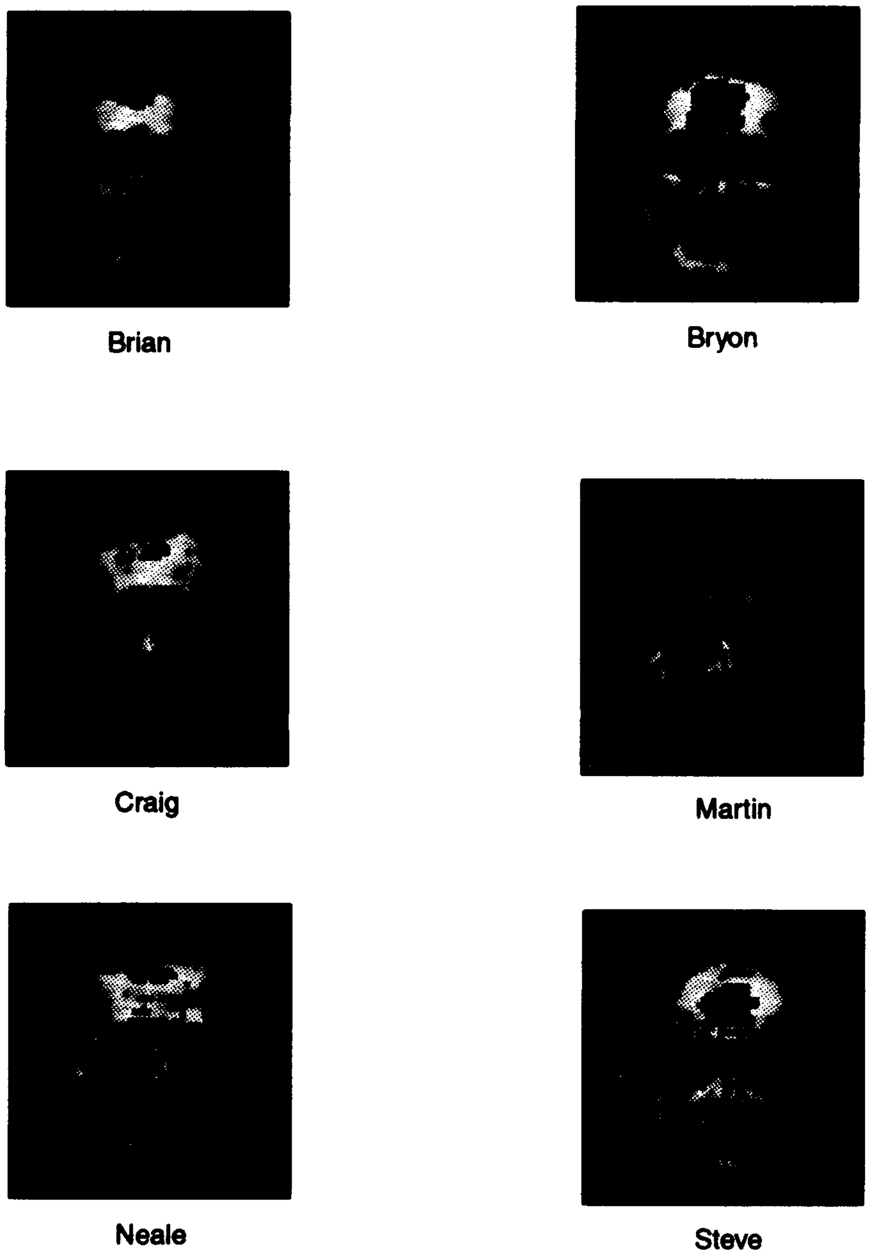


Figure 7. Six typical LBG face segmentations.

4.3 Neural Network

4.3.1 Faces. The same four training faces were used to train the LNKnet MLP routine, this time the RGB values of each pixel were used as the input vector instead of codebook values. Reconstructed segmented faces from this technique are shown in Figure 8. This did a much better job, no background has been misclassified as "face" and there are very few holes in heads.

To get an indication of how this would work in a cluttered background, images from an Atlanta Braves baseball game were segmented. The network was not retrained, a typical result is shown in Figure 9. The network did a very good job of pulling out face color, it even got faces in the crowd; however, quite a bit of background was misclassified. The errors were yellow or red, neither of these colors were represented in the training set. A new network was trained with yellow and red identified as background, the segmentation is also shown in Figure 9. Virtually no background has been misclassified, but the face reconstruction isn't as good, there are several holes in heads. Figure 10 shows a scatter plot of face pixels in hue and saturation space; hue is indicated by the angle in the polar plot, and saturation is indicated by the radius. Most pixels lie in the low saturation region around yellow and the reddish side of yellow; this indicates that keeping all face pixels means leaving some red and yellow background in the image; or conversely, eliminating all red and yellow, eliminates some face pixels.

4.3.2 Tanks. A new MLP was trained to look for tanks using the training image shown in Figure 11. Sections of tank turret and track were taken as class "0"; sections of grass, sky, and trees were taken as class "1" for LNKnet training purposes. The reconstructed training image in Figure 11 shows the network did very well, eliminating all the grass and sky. There was some problem with the trees, apparently they are too close to track color. "Two Tanks" and "Three Tanks" were presented to the network for segmentation, Figure 11 shows these results as well.

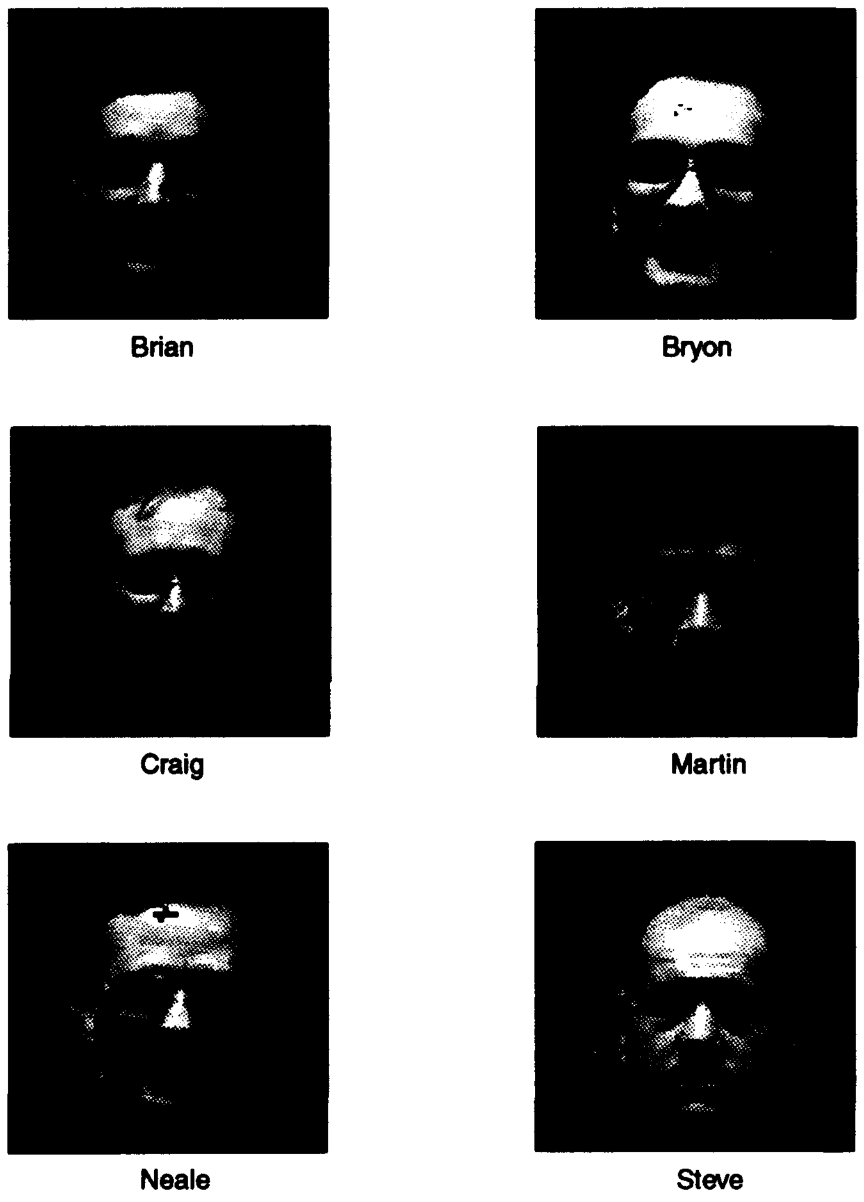
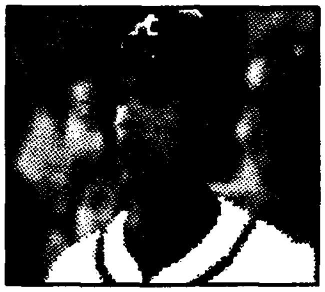
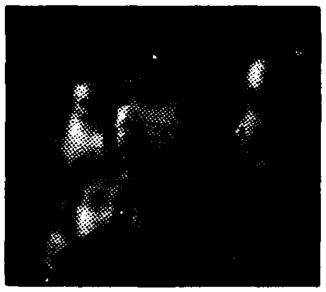


Figure 8. Six typical neural network face segmentations.


Original Image


1st segmentation

2nd Try

Figure 9. Segmentation with Cluttered Background.

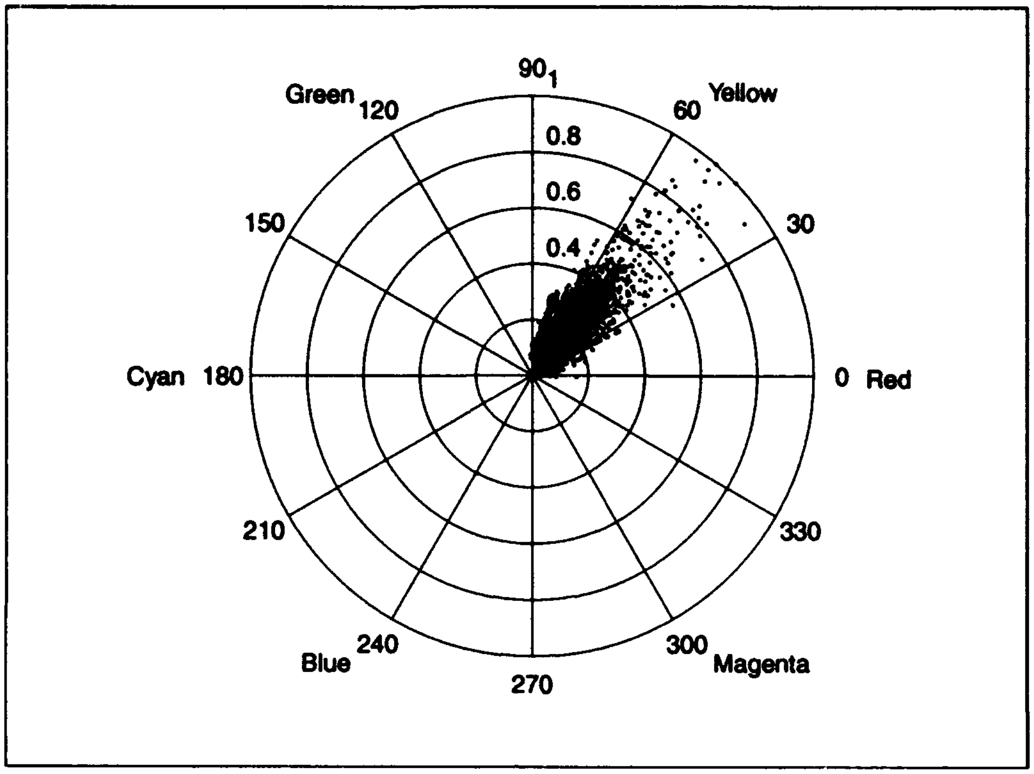
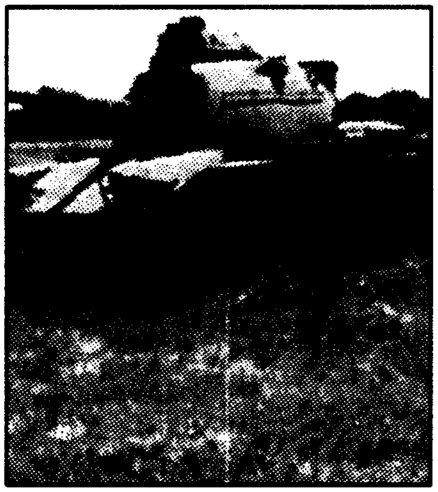
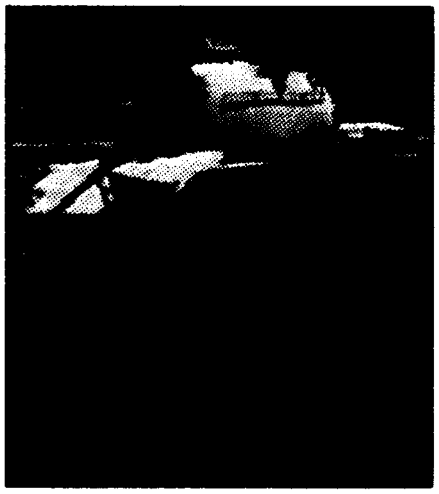


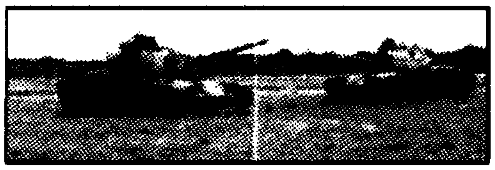
Figure 10. Scatter plot of face pixels.

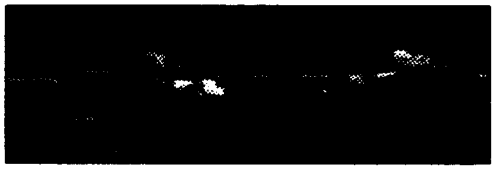
Again, the results are very good, but there is the same problem with the trees. "Three Tanks" shows a building in the segmented image. There were no buildings in the training image so the MLP could not learn to classify buildings as background.

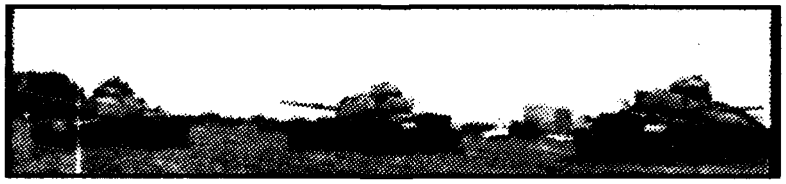
4.3.3 Airplanes. Yet another network was trained, this time to segment airplanes. Sections from frame numbers 1, 16, 32, 48, 64 were used to train the network. Frames 8 through 39 were used as the 32 frame sequence of moving data. These were chosen such that the plane fit in a 128 × 128 image over the whole 32 frames. Figure 12 shows some of the reconstructed planes. Several of the frames appear to be very noisy, the segmenter did not eliminate the sky very well; this was a problem with the frame grabbing system, as some of the frames were previewed before grabbing there was a noticeable change in sky color from time to time, sometimes the sky appeared quite blue and others it appeared very white. It was attempted


Original Training Image


Reconstructed Training Image


Original Two Tanks Image


Segmented Two Tanks


Original Three Tanks Image

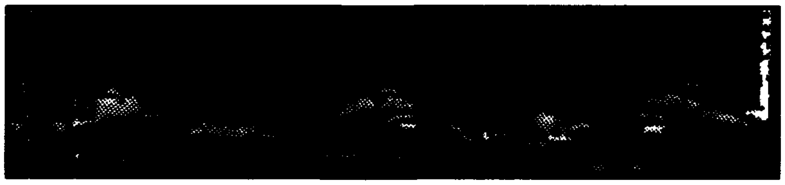

Segmented Three Tanks

Figure 11. Tank Segmentation.

to grab the frame at times when the sky was bluest, but these attempts were not always successful.

4.4 Heuristic Eye Finder

Having found faces with the neural network, the face recognition guys decided they needed eyes; a new network was developed which was trained such that eyes are not classified as part of the face. This left big holes where the eyes should be, examples of these reconstructions are shown in Figure 13. Figure 14 shows the results of the heuristic eye finder. The yellow dots identify line segments which could be part of an eye; the green dots identify cluster centers; the red dots identify the cluster centers which were determined to be eye positions. The images of Brian, Bryon, and Craig are all 256×256 pixels large; the images of Martin, Neale, and Steve are only 64×64 pixels large.

The eye finder worked regardless of the image size. There was some problem with Bryon's glasses. Figure 13 does not show the big holes in Bryon's eyes that are present in the other faces. Craig and Neale, who also wear glasses, did not have this problem; it is probably due to the way Bryon's head was tilted in the light.

There also seems to be a problem with Craig's right eye, the clusterer found two cluster centers in the eye (one green dot and one red dot). This is an artifact of the way the data was presented to the clusterer. The probable eye segments were simply presented sequentially, had their order been randomized this problem would have been avoided.

4.5 Wavelet "blob" detection

The Daubechies four tap filter was used for wavelet decomposition of tanks and airplanes. A two dimensional (x, y) decomposition was performed on the tanks, while a three dimensional (x, y, t) decomposition was performed on the sequence of airplanes. The 200×64 pixel image of "Two Tanks" was centered into a 256

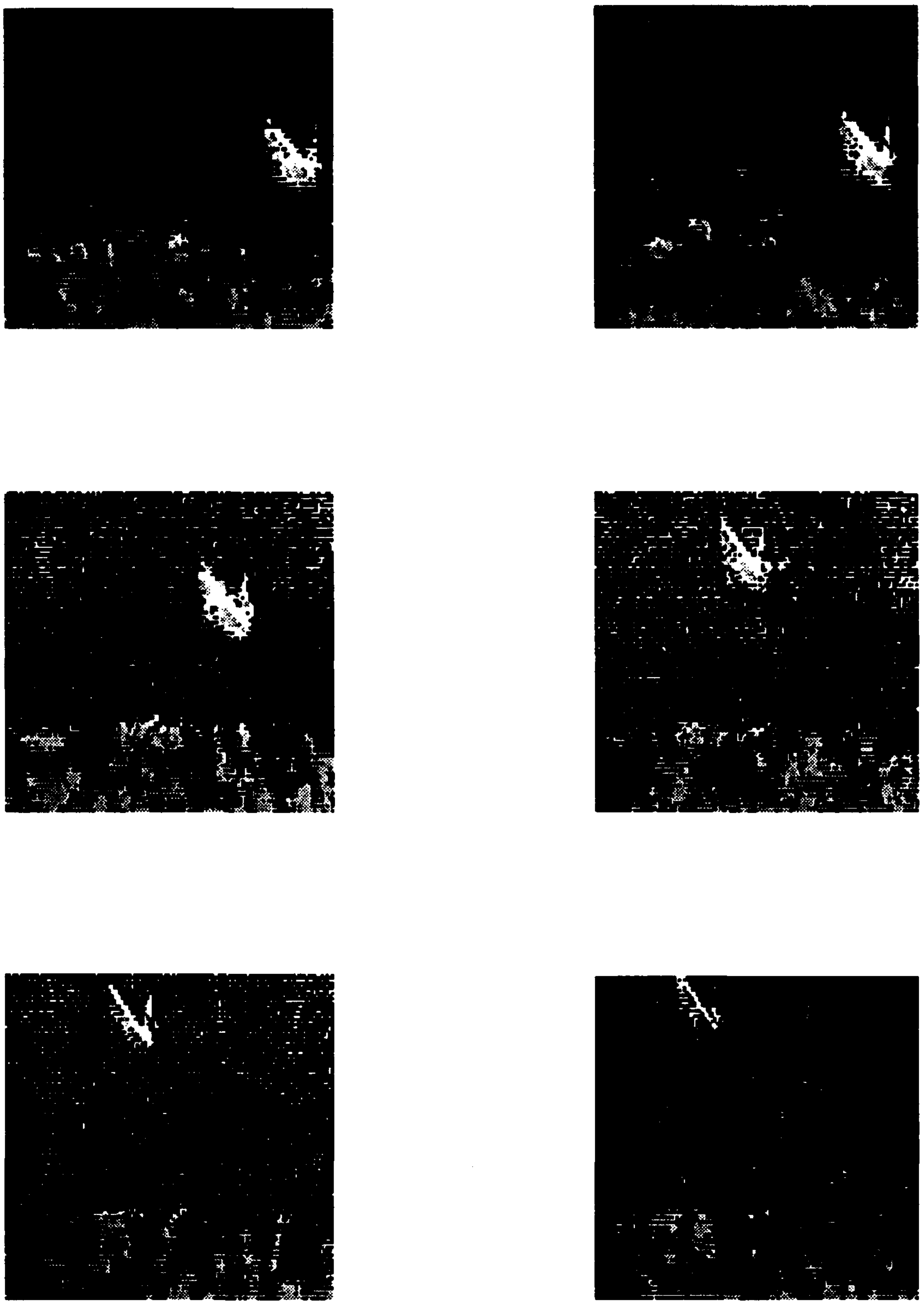


Figure 12. Airplane Segmentation.

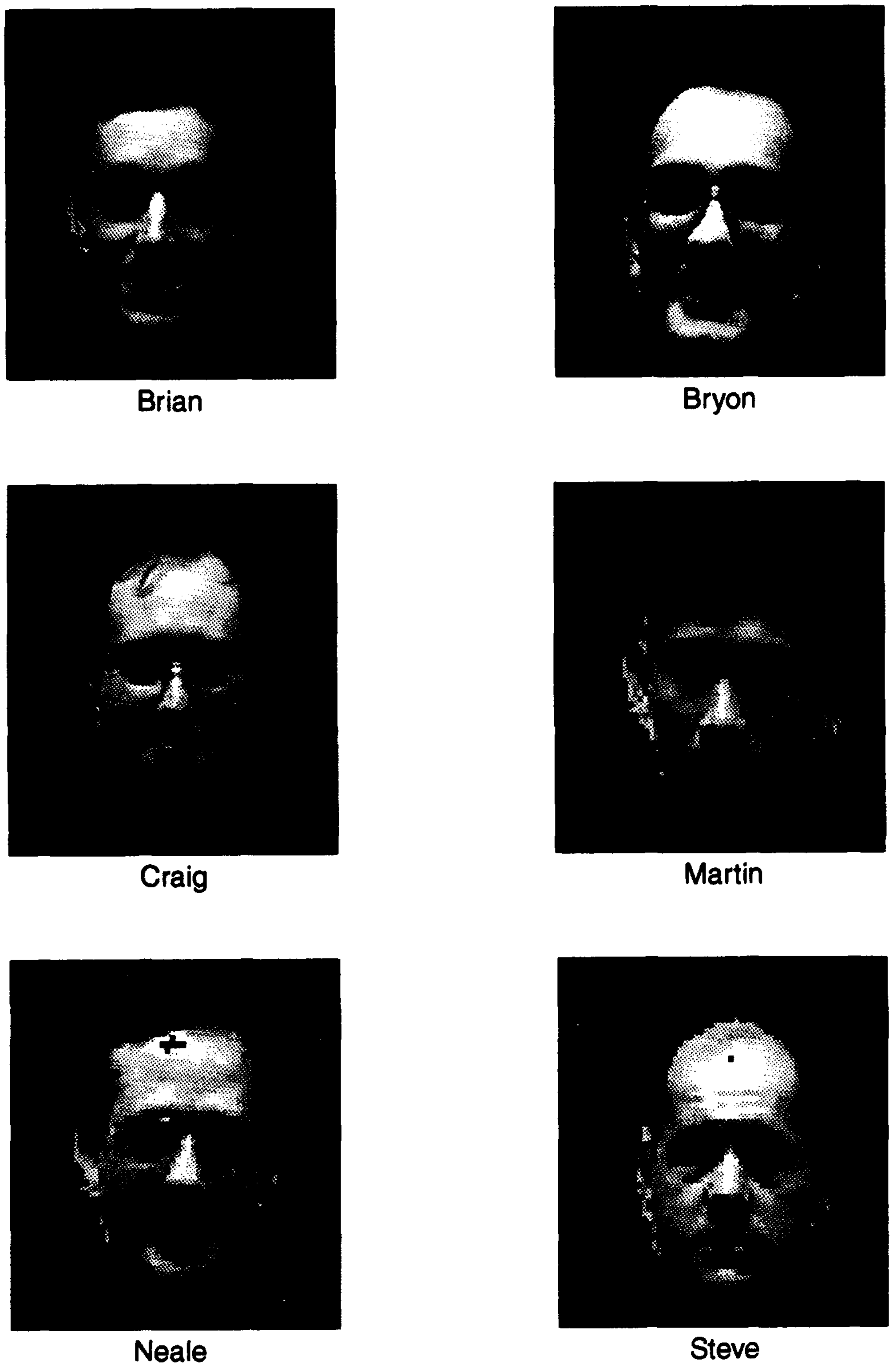


Figure 13. Faces with eyes segmented out.

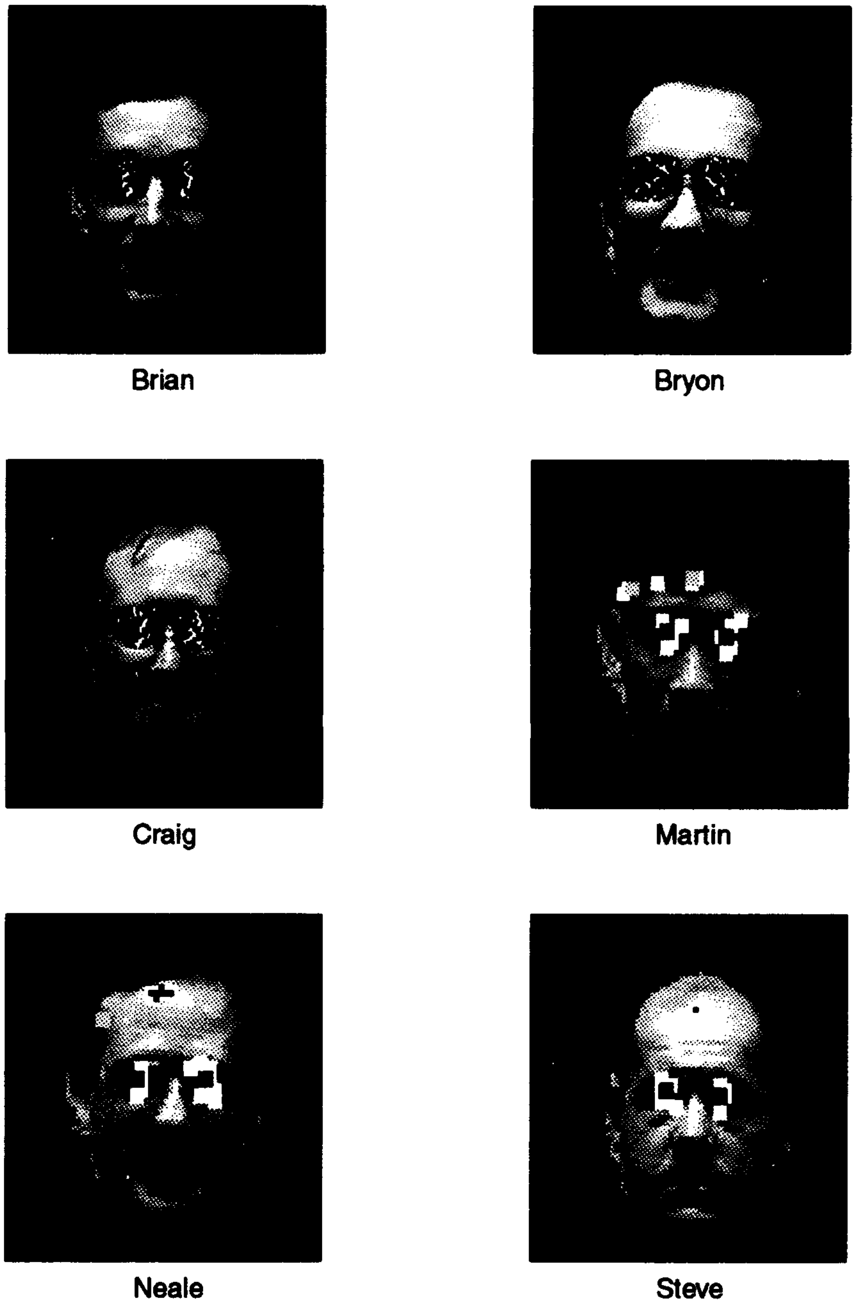


Figure 14. Eye positions as determined by the heuristic method. Yellow dots indicate possible eye segments, green dots indicate cluster centers, red dots indicate eye positions.

× 256 square image to accommodate the decomposition C code. Results of the decomposition of the unsegmented image (converted to gray scale) and the segmented image (binary pixels) are shown in Figures 15 and 16, respectively. No level of decomposition on the unsegmented image can distinguish the tanks. However, level four of the decomposition of the segmented image indicates tank by the white squares in both the approximation and details. Clearly, the color preprocessing segmentation was a great improvement.

The results from the three dimensional decomposition of the airplane sequence were not as promising. Figures 17 and 18 show the details for the second level spatial decomposition and the first level temporal decomposition for the unsegmented and segmented plane sequences respectively. The motion detector worked well on the original unsegmented data, the position of the plane is clearly indicated in the details. Unfortunately, the segmented details do not determine the plane position. The background of the unsegmented sequence is fairly uniform, there is no clutter for the plane to be lost in, there is little to be gained by the color segmentation. The effect of the segmentation was to add noise to the background and hide the plane in clutter in several of the frames of the sequence. A median filter over the segmented image before the wavelet decomposition would probably fix this problem. The combination of color and motion would probably be more useful on other types of problems, such as tracking a tank when it moves.

4.6 Correlation "blob" detection

Gabor filters were used to find eyes and tanks in the stationary images. The Gabor filter for finding eyes was centered at 8 with variance 36 and cosine frequency 1/16 in both the x and y directions. The Gabor filter for finding tanks was centered at (15,30) with variances 16, 36 and frequencies 1/30, 1/60 for the x and y directions respectively. Figures 19 and 21 show reconstructed objects with magenta "+" indicating the position of the object as determined by the correlator.

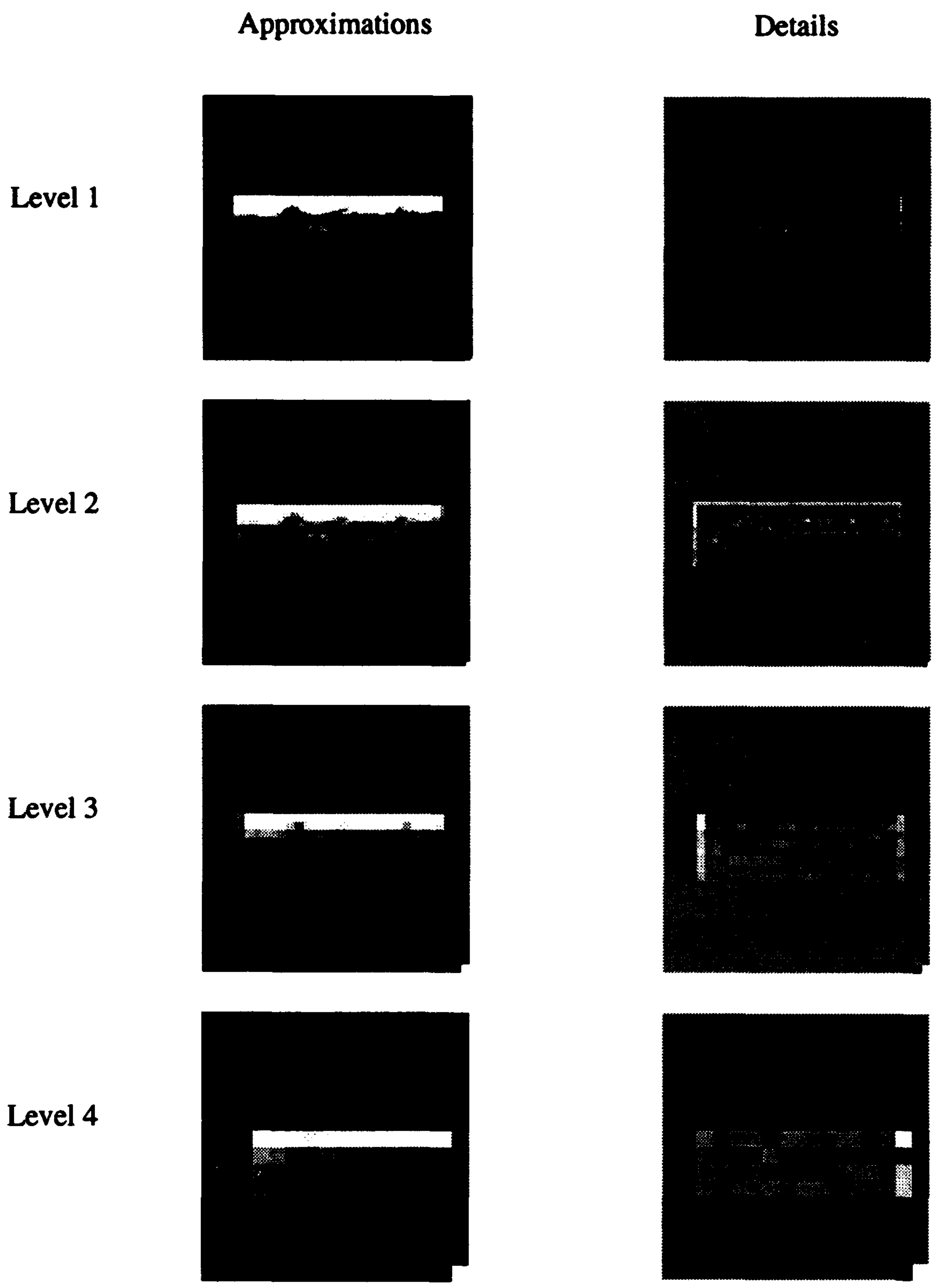


Figure 15. Approximations and details from decomposition of unsegmented image.

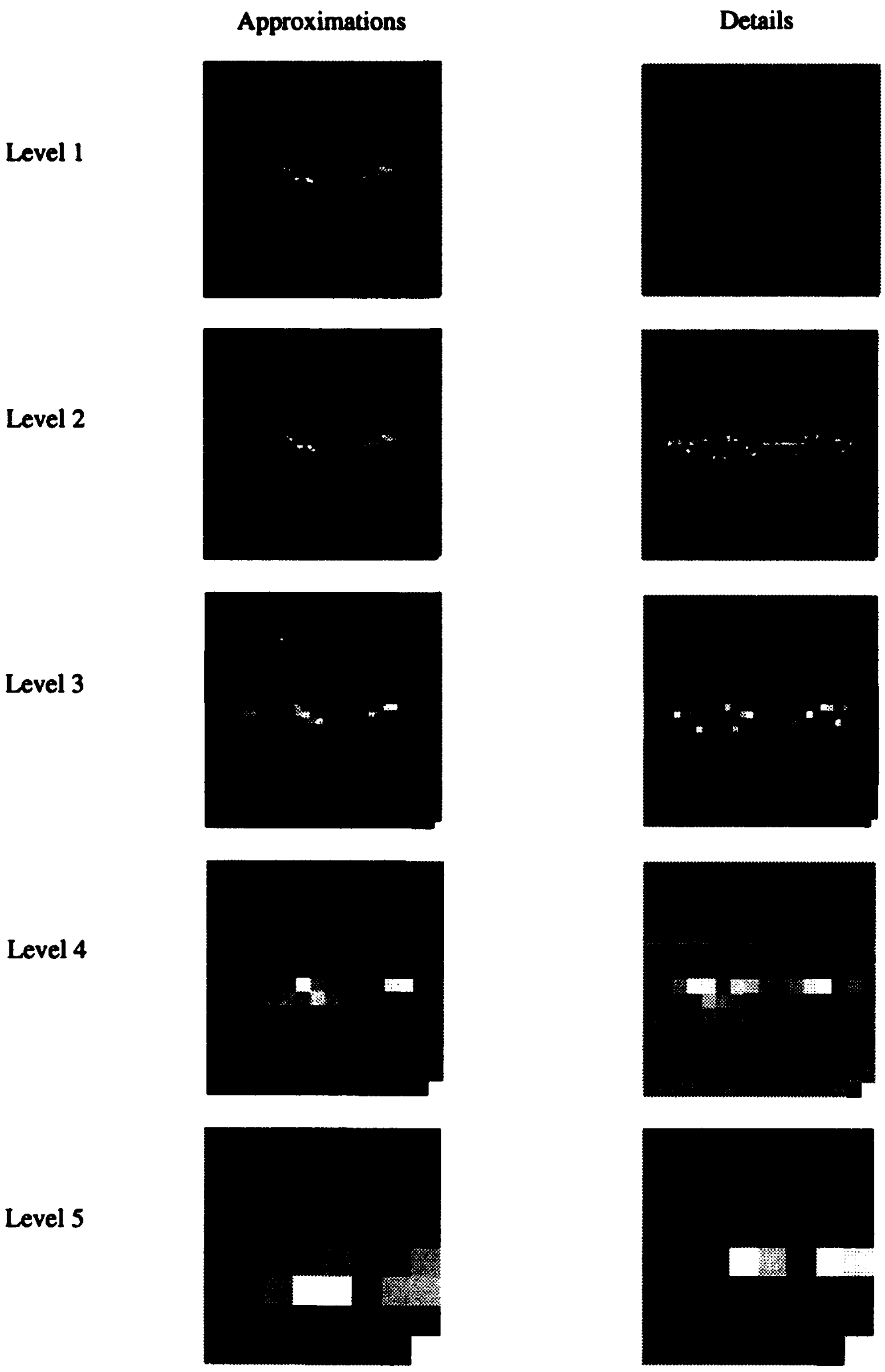


Figure 16. Approximations and details from decomposition of segmented image.

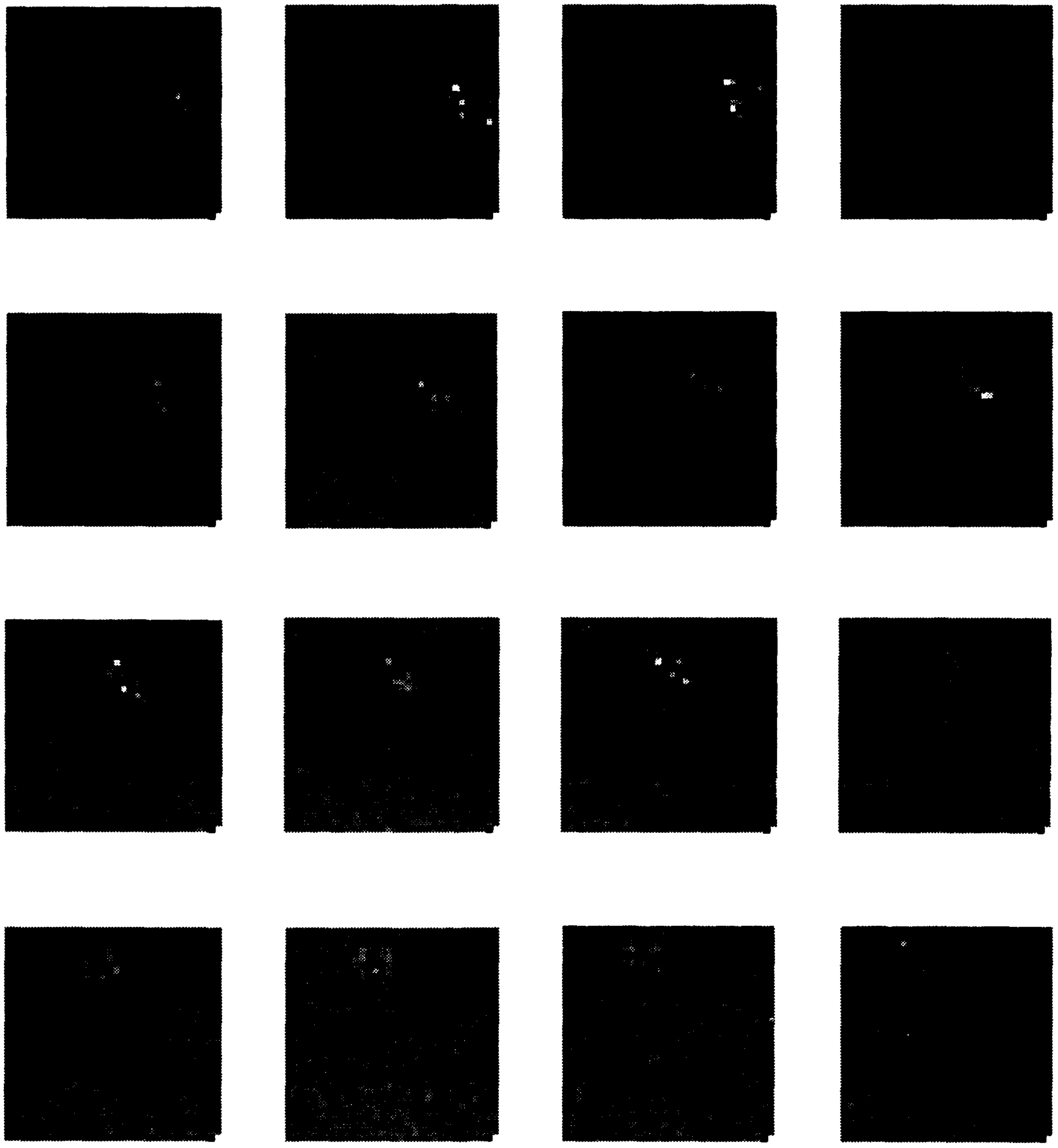


Figure 17. Details of spatial level 2, temporal level 1 decomposition of original image.

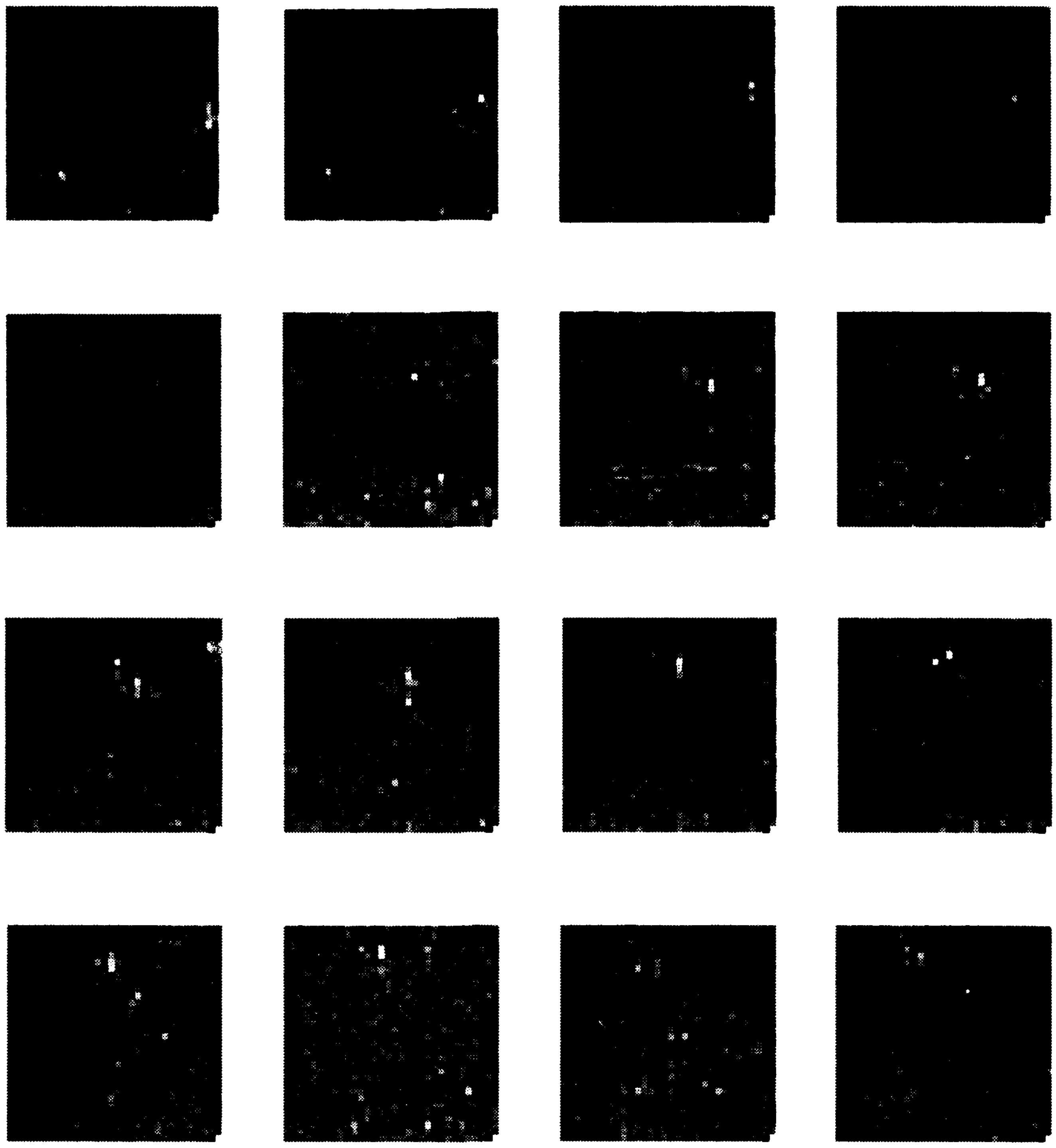


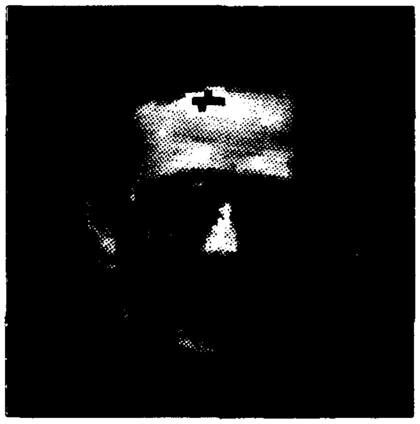
Figure 18. Details of spatial level 2, temporal level 1 decomposition of color segmented image.

The same filters were used on gray scale versions of the unsegmented images, Figures 20 and 21 show the results.

4.7 Conclusion

The color preprocessing step proved to be very helpful for finding eyes and tanks, it did not help find airplanes in poor images. The color segmenter seems to be most useful when the desired object is in a cluttered image (such as eyes and tanks) or when the object's color is clearly different from the background's color (faces). The color preprocessor actually hurt the segmentation results when the color quality of the image was poor, and the intensity image was adequate for segmentation.

Martin


Neale

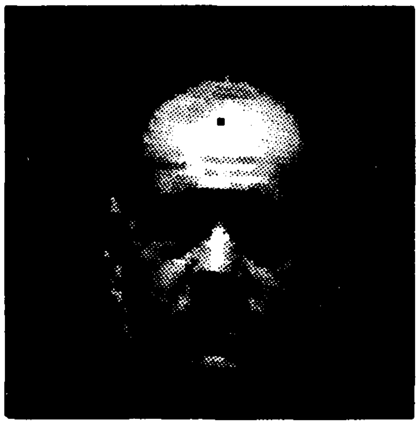
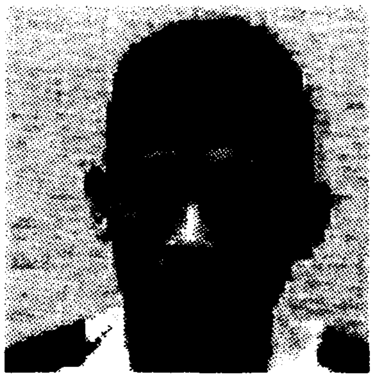
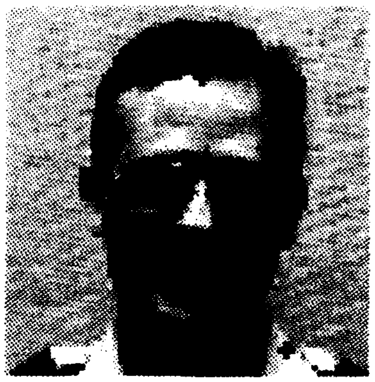

Steve

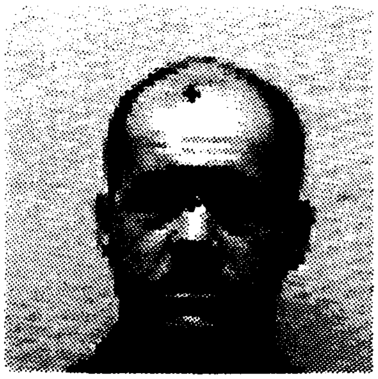
Figure 19. Eye positions as determined by the Gabor correlation method.



Martin



Neale



Steve

Figure 20. Eye positions as determined by the Gabor correlation method on the original image.

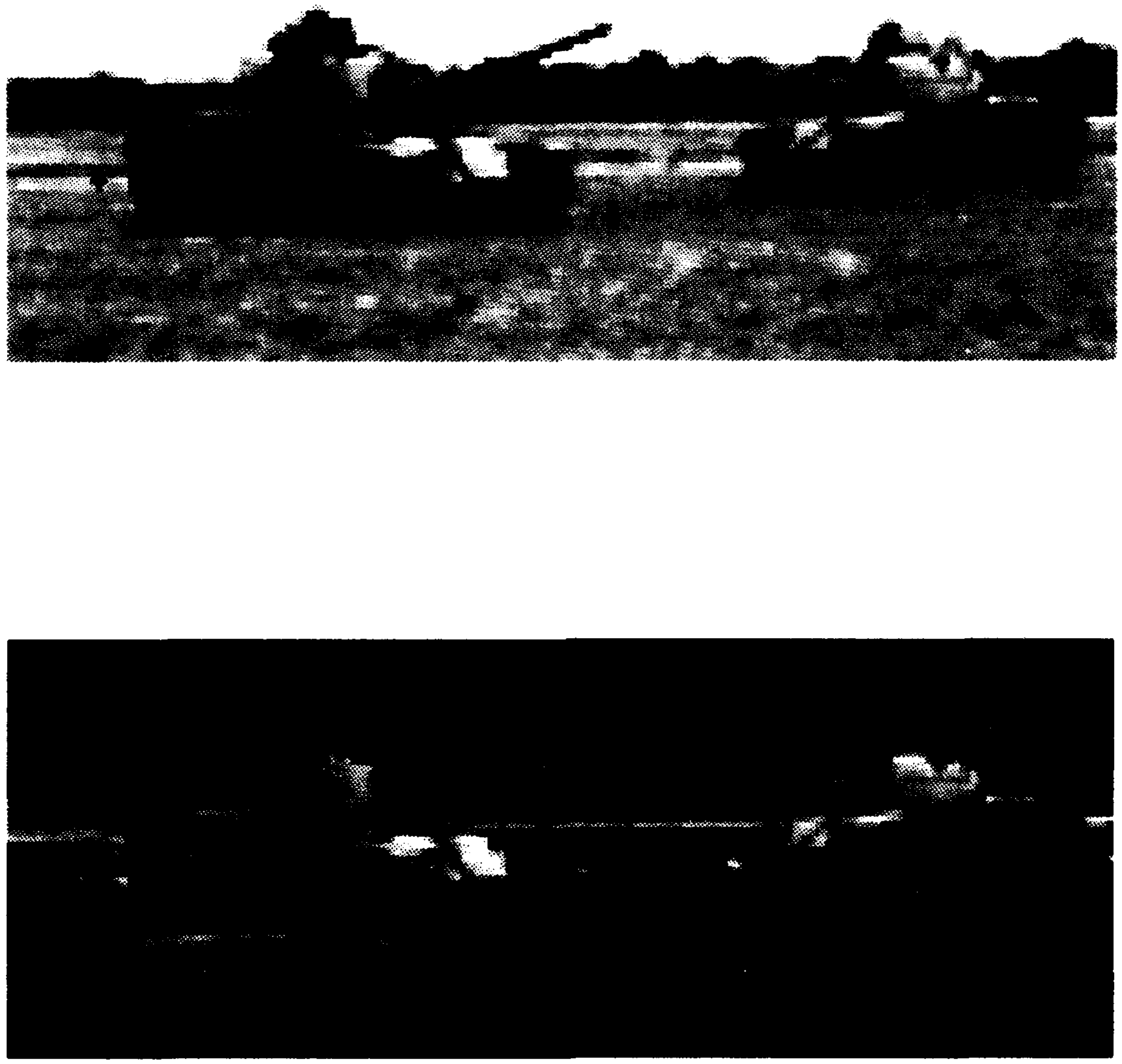


Figure 21. Tank position as determined by the Gabor correlator for both the original and color segmented images.

V. Conclusion

Color image segmentation has been demonstrated to be a useful first step in target recognition. The neural network color segmenter presented in this thesis gave good results. The network was able to learn face color and segment faces from both uniform and cluttered backgrounds. It was further able to separate eyes from the rest of the face based on color. Eye positions can now be determined and fed into a face recognition algorithm. Networks were also able to learn tank color and to segment tanks from background. The performance of a Gabor correlator and a wavelet decomposition was enhanced by using the segmented image instead of a gray scale version of the original image.

The fusion of color and motion detection was not completely successful, but it did demonstrate some potential. The data available for this experiment were segmentable based on motion alone; however, other problems where the target is not against a uniform background could take advantage of the color preprocessing step.

This was just a baby step into the color world. Future research could extend the neural network to not only find a particular color, but to determine what colors are present and segment them all. Other color coordinate systems could be used or combined with RGB. New detectors are being developed which can take a frame of data in many frequency bands. This type of data would lend itself to a three-dimensional wavelet decomposition were the axes are x, y, and color frequency. Further work could use a four-dimensional wavelet incorporating space, time, and color frequency decomposition.

Appendix A. C code

This appendix contains the important programs written for this thesis.

A.1 Reduce.c

```
/* reduce.c  This takes one rgb image and reduces its size prompting for desired size, resolution,
and offset  */

#include <stdio.h>
#include <stdlib.h>

typedef struct {unsigned char buff, blue, green, red;} colorpxl;

/************************************ Function make 2D matrix ****************************************
colorpxl **mkmatrix(int jmax, int imax){
   int i;
   colorpxl *stor;
   colorpxl **rval;

   rval = (colorpxl**)malloc(imax * sizeof(int));
   stor = (colorpxl*)malloc(imax * jmax * sizeof(colorpxl));
   for(i=0; i<imax; i++)  rval[i] = &(stor[i*jmax]);
return rval;}

/****************************************** Main Program **********************************************
main(int argc, char *argv[]){
   colorpxl **image, **newimage;
   int header[8], i, j, imax, jmax, smalli, smallj, ioffset, joffset, resx, resy, idx, jdx;
   char filename[10], inname[30], outname[30], ans[10];
   FILE *infile,*outfile;

/*** Set-up the I/O files ***/
   if(argc ≠ 2){
      printf("What color img file do you want to resize?\n");
      scanf("%s", &inname);}
   else
      strcpy(inname, argv[1]);
   i = 0;
   while(isalnum(inname[i])){
      outname[i] = inname[i];
      i++;}
   outname[i] = 0;
   strcat(outname, ".img");
```

```
    infile = fopen(inname,"rb");
    outfile = fopen(outname,"wb");

/*** Read in the original image ***/
    fread(header,4, 8, infile);
    imax = header[1]; jmax = header[2];
    image = mkmatrix(imax, jmax);
    fread(*image,sizeof(colorpxl),imax*jmax,infile);
    fclose(infile);

/*** Prompt for desired size, and allocate memory for the new image ***/
    smalli = 64; smallj = 64;
    printf("Default size is 64 by 64 is that OK? (y or n)\n");
    scanf("%s",ans);
    if(ans[0] ≠ 'y'){
        printf("what size would you like? (width height)\n");
        scanf("%d %d", &smalli, &smallj);
    }
    header[1] = smalli; header[2] = smallj;
    header[4] = 4 * smalli * smallj;
    fwrite(header, 1, 32, outfile);

    newimage = mkmatrix(smalli, smallj);

/*** Prompt for desired resolution ***/
    resx = 1;    resy = 1;
    printf("What resolutions would you like? (resx  resy)\n");
    scanf("%d  %d", &resx, &resy);

/*** Prompt for desired off set ***/
    ioffset = (imax - resx*smalli)/2;
    joffset = (jmax - resy*smallj)/2;
    printf("Default offsets are %d by %d is that OK? (y or n)\n", ioffset, joffset);
    scanf("%s",ans);
    if(ans[0] ≠ 'y'){
        printf("what offsets would you like? \n");
        scanf("%d %d", &ioffset, &joffset);
    }

/*** Create the reduced image ***/
    for(j=0; j<smallj; j++)
        for(i=0; i<smalli; i++)
            newimage[j][i] = image[resy*j + joffset][resx*i + ioffset];

/*** Write out the reduced image and close up ***/
    fwrite(*newimage, 1, header[4], outfile);
    fclose(outfile);
}
```

A.2 Reconstruct.c

```
/* reconstruct.c   This takes the error pixels from the mlp run on  the sparc and reconstructs the
face from just those pixels.                                                                    */

#include <stdio.h>
#include <stdlib.h>

typedef struct {unsigned char buff, blue, green, red;} colorpxl;

/********************************** Function make 2D matrix ******************************************
colorpxl **mkmatrix(int jmax, int imax){
   int i;
   colorpxl *stor;
   colorpxl **rval;

   rval = (colorpxl**)calloc(imax, sizeof(int));
   stor = (colorpxl*)calloc(imax * jmax, sizeof(colorpxl));
   for(i=0; i<imax; i++)  rval[i] = &(stor[i*jmax]);
return rval;}

/**************************************** Main Program **********************************************
main(int argc, char *argv[]){
   colorpxl **image, **wholeimage;
   int header[8], i, j, smalli, smallj, indx;
   char filename[10], inname[30], outname[30], oldname[30];
   FILE *infile,*outfile, *oldfile;

/*** Set-up the I/O files ***/
   if(argc ≠ 2){
      printf("what file do you want to use?\n");
      scanf("%s", &filename);}
   else
      strcpy(filename, argv[1]);
   sprintf(inname, "%s.err", filename);
   sprintf(outname, "%s.rec", filename);
   sprintf(oldname, "%s.img",filename);
   infile = fopen(inname,"r");
   outfile = fopen(outname,"wb");
   oldfile = fopen(oldname, "rb");

/*** Read in the original image and set up for the reconstructed image ***/
   fread(header,4, 8, oldfile);
   fwrite(header, 1, 32, outfile);

   for(i=0; i<8; i++) printf("%d\n", header[i]);

   smalli = header[1];  smallj = header[2];
```

```
    wholeimage = mkmatrix(smalli, smallj);
    image = mkmatrix(smalli, smallj);

    fread(*wholeimage,sizeof(colorpxl),smalli*smallj,oldfile);

/*** Read in the index of face pixels and reconstruct the image ***/
    while(!feof(infile)){
        fscanf(infile,"%d",&indx);
        j = (int)indx/smalli;
        i = indx%smalli;
        image[j][i] = wholeimage[j][i];
    }

/*** Write out the image and close up ***/
    fwrite(*image, 1, header[4], outfile);
    fclose(infile);
    fclose(outfile);
    fclose(oldfile);
}
```

A.3 Findeyes.c

```
/* findeyes.c        This takes the gray scale values of a recontructed image and finds probable eye
locations.
#include <stdio.h>
#include <stdlib.h>

/**************************************** Function make 2D matrix ****************************************
unsigned char **mkmatrix(int jmax, int imax){
    int i;
    unsigned char *stor;
    unsigned char **rval;

    rval = (unsigned char**)calloc(imax, sizeof(int));
    stor = (unsigned char*)calloc(imax * jmax, sizeof(unsigned char));
    for(i=0; i<imax; i++)  rval[i] = &(stor[i*jmax]);
return rval;}

/**************************************** Function make int 2D matrix ************************************
int **mkintmatrix(int jmax, int imax){
    int i;
    int *stor;
    int **rval;

    rval = (int**)calloc(imax, sizeof(int));
    stor = (int*)calloc(imax * jmax, sizeof(int));
    for(i=0; i<imax; i++)  rval[i] = &(stor[i*jmax]);
```

```
return rval;}

/******************************************** Main Program ******************************************
main(int argc, char *argv[]){
   unsigned char **image;
   float dist;
   int **hits, **eyes, *count, i, j, k, imax, jmax, maxdist;
   char inname[30], outname[30];
   FILE *infile,*outfile;

/*** Set-up I/O files ***/
   if(argc ≠ 2){
      printf("what file do you want to use?\n");
      scanf("%s", &inname);}
   else
      strcpy(inname, argv[1]);

   i = 0;
   while(isalnum(inname[i])){
      outname[i] = inname[i];
      i++;}
   outname[i] = 0;
   strcat(outname, ".I");

   infile = fopen(inname,"rb");
   outfile = fopen(outname,"w");

/*** Prompt for image size and set up arrays and read in image values ***/
   printf("how big is the square image?\n");
   scanf("%d", &imax);
   jmax = imax;

   image = mkmatrix(imax, jmax);
   hits = mkintmatrix(imax,jmax);
   eyes = mkintmatrix(2,jmax*5);
   count = (int*)calloc(jmax, sizeof(int));
   fread(*image,sizeof(unsigned char),imax*jmax,infile);
   fclose(infile);

/*** Find the indeces of pixels which are face ***/
   for(j=0; j<jmax; j++){
     k = 0;
     for(i=0; i<imax; i++)
      if(abs(image[j][i]) > 100){
        hits[j][k] = i;
        count[j]++;
        k++;
      }
```

```
    if(count[j] > 0)  count[j]--;        /* count[j] is the last index */
  }

/*** Look for blank segments in the top half of the image that are probable eye size ***/
  k = 0;
  for(j=jmax/4; j<5*jmax/8; j++){
    maxdist = hits[j][count[j]] - hits[j][0] + 1;
    i = 1;
    while(i ≤ count[j]){
      dist = hits[j][i] - hits[j][i-1];
      if((dist/maxdist > .1) && (dist/maxdist < .4) && (maxdist > imax/4)){
     eyes[k][1] = j;
     eyes[k][2] = (hits[j][i] + hits[j][i-1])/2;
        fprintf(tempfile, "%f   %d     %d\n", dist/maxdist, eyes[k][1], eyes[k][2]);
        fprintf(outfile, "\n%d     %d", eyes[k][1], eyes[k][2]);
     k++;
      }
      i++;
    }
  }
  printf("%d\n", k);
  fclose(outfile);
}
```

A.4 Clusterer.c

```
/* clusterer.c        This determines the eye position based on probable eye locations, it outputs an
image showing the recontructed face and the eye position.                                         */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define SQR(a) (a)*(a)

typedef struct {unsigned char buff, blue, green, red;} colorpxl;
typedef struct {float x, y; int ct;} clustcntr;

/************************************** Function make 2D matrix ****************************************
colorpxl **mkmatrix(int jmax, int imax){
  int i;
  colorpxl *stor;
  colorpxl **rval;

  rval = (colorpxl**)calloc(imax, sizeof(int));
  stor = (colorpxl*)calloc(imax * jmax, sizeof(colorpxl));
  for(i=0; i<imax; i++)  rval[i] = &(stor[i*jmax]);
```

```
return rval;}

/******************************************* Main Program *********************************************
main(int argc, char *argv[]){
   colorpxl **image, **wholeimage;
   clustcntr *center;
   int header[8], i, j, smalli, smallj, indx, m, n, size, ff, maxidx1, maxidx2, maxct1, maxct2;
   int clustct = 1, found;
   float thresh, dist, xdiff, ydiff;
   char filename[10], inname[30], outname[30], oldname[30], eyename[30], kname[30];
   FILE *infile, *outfile, *oldfile, *eyefile;

/*** Set-up I/O files ***/
   if(argc ≠ 2){
      printf("what file do you want to use?\n");
      scanf("%s", &filename);}
   else
      strcpy(filename, argv[1]);
   sprintf(inname, "%s.err", filename);
   sprintf(eyename, "%s.I", filename);
   sprintf(outname, "%s.eye", filename);
   sprintf(oldname, "%s.img",filename);
   sprintf(kname, "%s.try",filename);
   infile = fopen(inname,"r");
   outfile = fopen(outname,"wb");
   oldfile = fopen(oldname, "rb");
   eyefile = fopen(eyename, "r");

/*** Read in the old image, initilize everything ***/
   fread(header,4, 8, oldfile);
   fwrite(header, 1, 32, outfile);

   for(i=0; i<8; i++) printf("%d\n", header[i]);
   smalli = header[1];  smallj = header[2];
   wholeimage = mkmatrix(smalli, smallj);
   fread(*wholeimage,sizeof(colorpxl),smalli*smallj,oldfile);
   image = mkmatrix(smalli, smallj);

   printf("how big was the square image the eye positions came from?\n");
   scanf("%d", &size);
   ff = header[1]/size;
   thresh = size/10.0;
   center = (clustcntr*)calloc(100, sizeof(clustcntr));

/*** Reconstruct the face ***/
   while(!feof(infile)){
      fscanf(infile,"%d\n",&indx);
      j = (int)indx/smalli;
```

```
        i = indx%smalli;
        image[j][i] = wholeimage[j][i];
    }

/*** Put yellow dots where eyes might be and Find cluster centers ***/
    while(!feof(eyefile)){
      fscanf(eyefile,"%d  %d\n", &i, &j);
      for(m=i-1; m≤i+1; m++)
        for(n=j-1; n≤j+1; n++){
        image[m][n].red = 255;
        image[m][n].green = 255;
        image[m][n].blue = 0;
        }

      for(m=0, found = 0; (m<clustct) && (!found); m++)
        if(sqrt(SQR(center[m].x-j) + SQR(center[m].y-i)) < thresh){
           found = 1;
           center[m].x = (center[m].x*center[m].ct + j)/(center[m].ct+1);
           center[m].y = (center[m].y*center[m].ct + i)/(center[m].ct+1);
           center[m].ct++;
        }

      if(!found){
        center[m].x = j;
        center[m].y = i;
        center[m].ct = 1;
        clustct++;
      }
    }
    printf("%d\n", clustct-1);

/*** Put green dots on the clusters centers ***/
    for(i=1; i<clustct; i++){
      printf("%f  %f\n", center[i].x, center[i].y);
        for(m=(int)center[i].y-1; m≤(int)center[i].y+1; m++)
        for(n=(int)center[i].x-1; n≤(int)center[i].x+1; n++){
        image[m][n].red = 0;
        image[m][n].green = 255;
        image[m][n].blue = 0;
        }
    }

/*** Put red dots on the eye centers ***/
    maxct1 = center[1].ct;
    maxidx1 = 1;
    for(i=2; i<clustct; i++)
      if(center[i].ct > maxct1){
        maxct1 = center[i].ct;
      maxidx1 = i;
    }
      for(m=(int)center[maxidx1].y−1; m≤(int)center[maxidx1].y+1; m++)
      for(n=(int)center[maxidx1].x−1;n≤(int)center[maxidx1].x+1; n++){
    image[m][n].red = 255;
    image[m][n].green = 0;
    image[m][n].blue = 0;
      }

  if(maxidx1 ≠ 1){
    maxct2 = center[1].ct;
    maxidx2 = 1;
  }
  else{
    maxct2 = center[2].ct;
    maxidx2 = 2;
  }
  for(i=2; i<clustct; i++)
    if((center[i].ct > maxct2) && (i ≠ maxidx1)){
      maxct2 = center[i].ct;
      maxidx2 = i;
    }
      for(m=(int)center[maxidx2].y−1; m≤(int)center[maxidx2].y+1; m++)
      for(n=(int)center[maxidx2].x−1;n≤(int)center[maxidx2].x+1; n++){
    image[m][n].red = 255;
    image[m][n].green = 0;
    image[m][n].blue = 0;
      }

/*** Write out the new image and close everything up ***/
  fwrite(*image, 1, header[4], outfile);

  fclose(infile);
  fclose(outfile);
  fclose(oldfile);
  fclose(eyefile);
}
```

Appendix B. ESPS

This appendix contains the commands used for LBG clustering in the ESPS software package.

Add header information

```
        addfeahd -a conv_params file.dat file.fea
```

Create codebook

```
        vqdes -p vq_all_params file.fea file.cbk
```

Vector quantize a feature file

```
        vq -x3 -h histfile -f feature -c7 -i file.cbk file.fea file.vq
```

Extract codewords from file.vq

```
        pplain -f feature_cwndx file.vq > codewordfile.dat
```

To see the distortion results for each codebook level

```
        pplain -f final_dist file.cbk
```

Vita

Captain Kimberley A. McCrae was born on 6 June, 1963 in Lebanon, New Hampshire. She graduated from South Burlington High School in South Burlington, Vermont in 1981. She then attended the University of Vermont where she was awarded the degree of Bachelor of Science in Electrical Engineering in May of 1985. After her commissioning through OTS, she was assigned to the Air Force Weapons Laboratory, Kirtland AFB, Albuquerque, NM. There she worked as a Semiconductor Laser Engineer, characterizing two dimensional laser arrays. In 1989 she PCSed to the Ballistic Missile Organization, Norton AFB, San Bernadino, CA where she was Flight Test Project Manager for the Small ICBM program. In a desperate attempt to leave California she applied to, and was accepted by, AFIT as a Masters candidate in electro-optics. At AFIT she was co-founder of the "Ruck is my advisor support group". Though her advisor tried his best to prevent it, she did graduate in December, 1993. She has applied for the physics PhD program and may spend the next three years wishing she was back in California.

Permanent address: 49 Orchard Road
South Burlingtion, Vt 05403

Bibliography

1. Burns, Thomas J. *A Non-Homogeneous Wavelet Multiresolution Analysis and its Application to the Analyis of Motion*. PhD dissertation, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1993.

2. Burns, Thomas J. and others. "Spatio-temporal signal processing using a discrete 3D multiresolution wavelet analysis," *SPIE Proceedings Intelligent Robots & Computer Vision XI: Biological, Neural Net, and 3D Methods, 1826* (November 1992).

3. Carlotto, Mark J. "Histogram Analysis Using Scale-Space Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-9* (January, 1987).

4. Celenk, Mehmet. "A Color Clustering Technique for Image Segmentation," *Computer Vision, Graphics, and Image Processing, vol. 52* (1990).

5. Fielding, Kenneth and others. "Spatio-temporal Pattern Recognition using Hidden Markov Models." *Proceedings of the SPIEvol. 2032*. 1993.

6. Gay, Kevin P. *Autonomous Face Recognition*. MS thesis, Air Force Institute of Technology, 1992.

7. Gray, Robert and others. *Locally Optimal Block Quantization for Sources without a Statistical Model*. Technical Report L-904-1, Stanford, CA: Stanford University Information Systems Lab, May 1979.

8. Huang, Chung-Lin and others. "Color Images' Segmentation Using Scale Space Filter and Markov Random Field," *Pattern Recognition, vol. 25* (1992).

9. Keller, John G. *Identity Verification Through the Fusion of Face and Speaker Recognition Techniques*. MS thesis, Air Force Institute of Technology, 1993.

10. Kim, Jeong-Yeop and others. "Color Image Enhancement Based on Modified IHS Coordinate System." *SPIE Intelligent Robots and Computer Vision XI vol. 1825*. 366–377. 1992.

11. Krepp, Dennis L. *Face Recognition with Neural Networks*. MS thesis, Air Force Institute of Technology, 1992.

12. Ledley, R. S. and others. "Fundamentals of True-color Image Processing." *IEEE Proceedings*. 791–795. 1990.

13. Linde, Yoseph and others. "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications, vol. com-28* (1980).

14. Mallat, Stephane G. "A Theory for Multiresolution Signal Decomposition: the Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11* (July, 1989).

15. Nakauchi, Shigeki and others. "Analysis of Munsell Color Space by a Five-layered Neural Network." *Annual International Conference of the IEEE Engineering in Medicine and Biology Societyvol. 1699*. 1419–1420. 1990.

16. Rogers, Steven K. and others. *An Introduction to Biological and Artificial Neural Networks*. Bellingham, WA: SPIE Optical Engineering Press, 1991.

17. Scharcanski, J. and others. "Color Texture Representation Using Multi-scale Feature Boundaries." *SPIE Visual Communications and Image Processingvol. 1818*. 156–163. 1992.

18. Tou, J. T. and R. C. Gonzalez. *Pattern Recognition Principles*. Reading, MA: Addison-Wesley Publishing Company, 1974.

19. Zeki, Semir. "The Visual Image in Mind and Brain," *Scientific American* (September, 1992).

REPORT DOCUMENTATION PAGE	Form Approved OMB No 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1993	Master's Thesis

4. TITLE AND SUBTITLE COLOR IMAGE SEGMENTATION	5. FUNDING NUMBERS
6. AUTHOR(S) Kimberley A. McCrae	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB OH 45433-6583	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GEO/ENG/93D-02
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dr. Rodney Winter DIR NSA/R22 9800 Savage Road Fort Meade, MD 20755	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Distribution Unlimited	

13. ABSTRACT *(Maximum 200 words)*

The most difficult stage of automated target recognition (ATR) is segmentation. Current AFIT segmentation problems include faces and tactical targets; previous efforts to segment these objects have used intensity and motion cues. This thesis develops a color preprocessing scheme to be used with the other segmentation techniques. A neural network is trained to identify the color of a desired object, eliminating all but that color from the scene. Gabor correlations and 2D wavelet transformations will be performed on stationary images; and 3D wavelet transforms on multispectral data will incorporate color and motion detection into the machine visual system. The thesis will demonstrate that color and motion cues can enhance a computer segmentation system. Results from segmenting faces both from the AFIT data base and from video taped television are presented; results from tactical targets such as tanks and airplanes are also given. Color preprocessing is shown to greatly improve the segmentation in most cases.

14. SUBJECT TERMS Color, Segmentation, Neural Network, Wavelet Transform	15. NUMBER OF PAGES 65
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL